Liikkeenhallinta tietokoneanimaatiossa

Simo Fält

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelytieteiden tutkintoohjelma
Pro gradu -tutkielma
Ohjaaja: Martti Juhola
Marraskuu 2015

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelytieteiden tutkinto-ohjelma
Fält, Simo: Liikkeenhallinta tietokoneanimaatiossa
Pro gradu -tutkielma, 54 sivua
Marraskuu 2015

Tänä päivänä animoituja kuvia näkyy kaikkialla. Lyhyiden animaatioiden luominen on huomattavasti helpompaa ja halvempaa kuin vaikka viisitoista vuotta sitten. Animaatiosta on tullut oleellinen osa laitteiden käytettävyyttä ja visuaalisuutta. Animaatiolla voidaan myös vastata elokuvateollisuuden alati kasvaviin vaatimuksiin.

Tässä tutkielmassa käsitellään liikkeenhallintaa tietokoneanimaatiossa. Tutkielma on katsaus merkittävimpiin tutkimuksiin tietokoneella mallinnetusta animaatiosta. Tutkielmassa käydään lävitse yleisimmin käytetyt tavat luoda ja muokata liikettä tietokoneella.

Avainsanat ja -sanonnat: animaatio, liikkeenhallinta, kinematiikka, liikkeenkaappaus.

Sisällys

1 Johdanto

Useimmalle meistä tietokoneanimaatiosta tulee mieleen animaatioelokuva. Tietokoneella animoitu kuva on nykypäivänä kuitenkin arkinenkin asia, jota moni ei tule ajatelleeksi. Älypuhelimen ruudulla vilkkuva kirjekuori saapuneen sähköpostin merkkinä on animaatio vaikkakin yksinkertainen. Kun kuvaketta koskee viestiä avatakseen, seurauksena on usein lyhyiden animaatioiden sarja. Sileän lasipinnan antama vaste sille, että sormi on varmasti onnistuneesti osunut kirjekuoren kuvaan, saattaisi olla valju ilman kirjekuoren pientä liikettä, joka saa aikaan vaikutelman siitä, että kirjekuori liikkuu vähän kauemmas pinnasta. Avautuva viesti ei myöskään ilmesty tyhjästä täyttämään koko näyttöä. Sen sijaan viesti näyttää kasvavan pienestä ikonista lopulliseen mittaansa, ehkä jopa niin että viestin osuminen ruudun keskelle saattaa vaatia ylimääräisiä keskitysyrityksiä. Pienten animaatioiden ansioista erilaisiin laitteisiin on saatu paremman käytettävyyden lisäksi silmänruokaa ja näyttävyyttä. Esimerkin kirjekuoresta kasvavalla viestillä ja animaatiofilmillä on paljon yhteistä. Algoritmi, jolla kuvaan haluttava liike lasketaan, saattaa hyvinkin olla sama molemmissa tapauksissa.

Yksinkertaisimmillaan animaatio lienee kaksi toisestaan poikkeavaa kuvaa, joita nopeasti vuoronperään näytettäessä saa aikaan illuusion liikkeestä. Se, kuinka toisistaan poikkeavat kuvat luodaan, voi vaihdella. Tässä tutkielmassa käydään lävitse erilaisia luontitapoja. Tavasta tuottaa noita kuvia käytetään termiä liikkeenhallinta. Osa kuvatuista menetelmistä muistuttavat hyvin paljon menetelmiä, joita käytetään, kun animaatioita luodaan puhtaasti käsin.

Tutkielmassa käydään lävitse liikkeenhallintaa vain yleisen yksinkertaisen kappaleen kannalta, mutta myös monimutkaisemman nivelikkään olion, kuten ihmisen, kannalta. Tutkielmasta on jätetty ulkopuolelle erilaiset erikoistapaukset, kuten erilaisten kankaiden tai pistejoukkojen animointi. Tutkielmassa yritetään avata kunkin menetelmän hyviä puolia mutta myös kyseisten menetelmien ongelmia. Liikkeenhallinnassa ei ole olemassa yhtä oikeaa tapaa hallita simuloitavaa liikettä.

Tämän tutkielman luvussa 2 esitellään animaation historia. Luvussa 3 käsittellään perinteistä kelmuanimaatiota muistuttavaa avainkuva-animointia, jossa tietokone laskee tarvittavat välikuvat avainkuvien välille. Luku 4 esittelee etenkin sitä seuraavissa luvuissa käytetyn tietorakenteen. Luvussa 5 käsitellään kinemaattiset menetelmät

liikkeenhallinnassa. Luvun aluksi käsitellään suora- ja käänteiskinematiikka sekä siihen liittyvät yleisimmät ratkaisumenetelmät. Luvun loppu antaa pikaisen katsauksen liikkeenhallintaan, joka koostuu saman lyhyemmän sekvenssin toistosta. Luvussa 6 käsitellään dynaamisia liikkeenhallinta menetelmiä. Lisäksi luvussa käsitellään dynamiikan kannalta oleellista kappaleiden välisiä törmäyksiä sekä erilaisia tapoja hallita ja rajoittaa dynaamisin menetelmin tuotettua liikettä. Luvussa 7 käsittellään liikkeenkaappausta ja luvussa 8 tuotetun liikkeen muokkausta.

2 Historia, kelmusta pikseleihin

Animaation historia juontaa juurensa aina 1800-luvulle. Ihmistä kiehtoi ajatus esittää liikettä, johon ainoastaan hetken tallentanut valokuva ei pystynyt. Erilaisilla peili- ja linssilaitteilla pystyttiin heijastamaan esimerkiksi seinälle joukko metsästäjiä villipedon perässä tai laukkaava hevonen. Näiden esitysten lyhyt kesto tuntuu tämän päivän kokoillan elokuviin verrattuna mitättömältä, mutta kun ajatellaan sen ajan tuotantovälineitä, niin tilanne onkin toinen. Jokainen kuva jouduttiin piirtämään tai kuvaamaan alusta loppuun käsin. Alkuun jokainen kuva oli myös oma kokonaisuutensa, niin liikkuvan kohteen kuin myös taustojen suhteen. Vasta myöhemmin oivallettiin, että kuvaan voitiin liittää myös staattisia elementtejä ja piirtää yhä uudestaan ainoastaan liikkeeseen vaikuttavat kohdat. Siirtyminen valkoiselta paperilta läpinäkyvälle kelmulle saattoi tuntua jostain animaattorista samanlaiselta edistysaskelelta kuin siirtyminen tällaisesta kelmuanimaatiosta kokonaan tietokoneella tuotettuun animaatioon.

Tietokoneen käyttö animaation teossa on nostanut rimaa animaation visuaalisuuden suhteen. Se on mahdollistanut uudenlaisten kuvien luomisen, joka käsin olisi vaatinut liian suuren työmäärän. Lisäksi on huomattu, että tietokoneella saatetaan tuottaa animaatioita, jotka ovat niin realistisen näköisiä, että niitä voidaan käyttää hyväksi ihmisnäyttelijöiden rinnalla. Tietokoneen avulla voidaan elokuvaan tuoda kalliita tai jopa mahdottomalta tuntuvia elementtejä, kuten realistisen näköisiä dinosauruksia. Tietokoneen käyttö animaation teossa on mahdollistanut sen, että raja todellisen ja keinotekoisesti tuotetun välillä on hämärtynyt lähes huomaamattomaksi. Tietokoneella tuotettu animaatio vähintään efektien muodossa onkin arkipäivää lähes jokaisessa nykypäivänä tuotetussa elokuvassa. Joissain elokuvissa käytetään jopa niin paljon tietokoneella tuotettua animaatioita, että on vaikea sanoa, pitäisikö puhua enää tietokone-efektein varustellusta elokuvasta vai näyttelijöin terästetystä animaatiosta. Hyvänä esimerkkinä elokuvasta voidaan mainita vuoden 2007 *Beowolf* [BEO], jossa kaikkien hahmojen taustalla on oikea näyttelijä, jonka piirteet ovat selvästi havaittavissa hahmoissa. Tietokoneanimaation viimeisin aluevaltaus lienee reaaliaikainen animointi, jossa animaation liikettä hallitaan reaaliaikaisesti erilaisten liikkeenkaappaustekniikoiden kautta. Tämä on yleistynyt viime vuosina etenkin pelikonsoleissa, joissa käytetyt liikeradat ovat kuitenkin rajoitettu valmiiksi animoituihin liikkeisiin.

Katsottaessa viime vuosina tehtyjä tietokoneanimaatioita, jopa kokoillan elokuvia, ne häikäisevät realistisuudellaan. On helppoa kyseenalaistaa aitojen näyttelijöiden tarpeellisuus. Usein keskustelua herättääkin nimenomaan ihmisten tai ihmisen kaltaisten hahmojen realistisuus. Tietokoneella luodut räjähdykset ja erikoistehosteet ovat olleet arkipäivää jo pitkään. Monikaan ei kuitenkaan tule ajatelleeksi sitä, kuinka paljon hahmon realistisuuteen vaikuttaa pienet seikat, esimerkiksi hahmon vaatteiden ja hiusten liikkeet sekä nivelten liikeradat. Näiden asioiden kuvantamiseen on kehitetty erilaisia algoritmeja ja työkaluja helpottamaan animaattorin työtä. Usein ongelmana on, että se mikä toimii pomppivan pallon animaation luomiseen, ei välttämättä sovellu ihmisen animointiin tai päinvastoin. Erilaisista tavoista hallita ja kuvata liikkeitä käytetään tässä tutkielmassa käsitettä liikkeenhallinta.

Animoitaessa ihmisen liikettä puhutaan usein liikkeen simuloinnista. Menetelmät, joilla tuotetaan simuloitua liikettä, voidaan karkeasti jakaa kolmeen kategoriaan: avainkuva-animointi, kinemaattiset ja dynaamiset menetelmät. Näiden lisäksi käytetään liikkeenkaappausta, jossa liike luodaan aitojen näyttelijöiden avulla.

Avainkuva-animaation voidaan katsoa pohjautuvan perinteiseen tapaan tehdä animaatioita. Perinteisessä animaatiossa animaattori luo sarjan kuvia, joissa jokainen kuva on osa liikettä. Tällainen animointi on työlästä, koska jokainen asento tulee kuvata erikseen. Staattisten kuvien luomaan liikkeen illuusioon vaaditaan kuitenkin vähintään 12 kuvaa jokaista sekuntia kohti. Yleisin kuvataajuus animaatiossa lienee kuitenkin 24 FPS (Frames Per Second), johon päästään 12 kuvallakin käyttämällä jokaista kuvaa kaksi kertaa peräkkäin. Taajuuden 24 FPS vaatimus juontaa juurensa elokuvan alkuajoilta, jolloin kuvataajuudeksi vakiintui 24 kuvaa sekunnissa. Animaatiota luotaessa pitääkin miettiä mahdollista jakelukanavaa ja sen asettamia vaatimuksia kuvataajuudelle.

Toisin kuin perinteisessä läpinäkyvälle kelmulle piirretyssä avainkuva-animaatiossa, tietokoneanimaatiossa animaattorin tehtäväksi jää yleensä kuvata liikkeen alkutilanne ja lopputilanne, jonka jälkeen erilaisilla algoritmeilla lasketaan liikerata ja kappaleiden asennot eri ajanhetkellä. Kineettisesti mallinnettu animaatio onkin ylivertainen mallintamistapa, jos ajatellaan liikkeen tuottamisen helppoutta tai metodin ymmärtämisen helppoutta. Mallintamisen huonous näkyy siinä, että vähänkin vaikeamman kappaleen animointi kinemaattisesti on vaikeaa. Jo ihmisen liikkeiden realistinen mallintaminen on lähes mahdotonta tai vähintään hyvin työlästä.

Dynaamisella simuloinnilla liikkeenhallinnan yhteydessä tarkoitetaan menetelmiä, joissa kappaleen ja ympäristön fyysiset ominaisuudet tunnetaan. Metodit laskevat liikkeen automaattisesti esimerkiksi kappaleeseen kohdistetun voiman ja kappaleen painon perusteella. Dynaamiset metodit noudattavat usein fysiikan lakeja ja etenkin realistisissa animaatioissa tuottavat hyviä tuloksia. Dynaamiset hallintamenetelmät ovat myös huomattavasti yksinkertaisempia kuin kinemaattiset menetelmät, kun animoidaan useita kappaleita samanaikaisesti. Yhtenä esimerkkinä voidaan ajatella lentävän lintuparven animointia. On huomattavasti helpompaa kertoa parven koko ja yhden linnun fysikaaliset ominaisuudet kuin animoida pienenkin parven lintujen yksittäiset liikeradat. [Watt, 2000]

3 Avainkuva-animointi

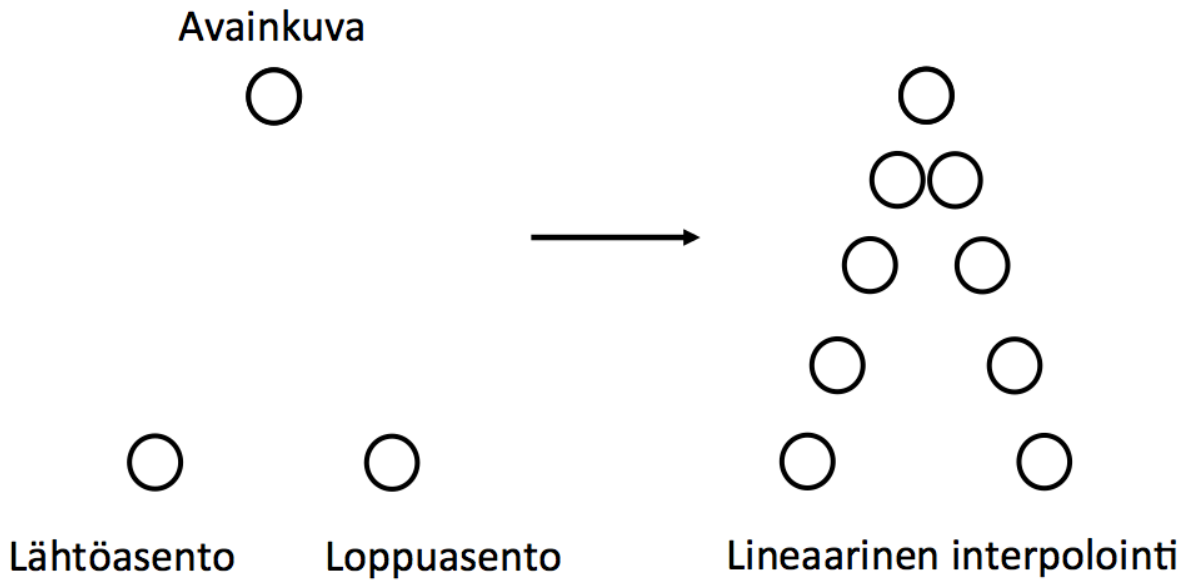
Kiinteän kappaleen animointi on yksinkertaisin animaation muoto. Siitä on tullut niin olennainen osa nykypäivää, että harva ihminen tulee kiinnittäneeksi siihen enää suurta huomiota. Kiinteän kappaleen animointi on yksinkertaisimmillaan katselukulman siirtoa paikasta toiseen, jolloin katsoja näkee ruudussa liikkuvan kappaleen. Esimerkkinä tällaisesta voisi olla tv-mainoksessa lentävä yrityksen logo. Pelkän katselukulman siirron lisäksi myös itse kappale voi liikkua oman akselinsa ympäri. Kappaletta voidaan liikuttaa myös virtuaalisessa maailmassa tai katselukulman ja koordinaatiston yhdistelmässä. Tällainen on tarpeen esimerkiksi animoitaessa näkymää, jossa on useita kappaleita, jotka liikkuvat suhteessa toisiinsa. Kiinteän kappaleen animoinnissa käytettyjä tekniikoita ovat avainkuva-animointi (keyframe) ja interpolointi. Avainkuva-animointi muistuttaa hyvin paljon perinteistä kelmuanimointia, jossa pääanimaattori luo liikkeen päälinjat määrittämällä liikkeen äärikohdat. Pääanimaattorin tehtävänä on luoda vain ne kuvat, joissa liikkeen suunta muuttuu ja määritellä kuvien välillä kuluva aika. Pääanimaattorin lisäksi on usein muita animaattoreita, joiden tehtäväksi jää täyttää näiden pääanimaattorin luomien avainkuvien välinen aika, jotta nopeasti näytettävä kuvasarja esittää liikkeen halutulla tavalla. Tietokoneanimaatiossa pääanimaattorilla on sama vastuu, mutta välikuvien luonti jätetään tietokoneen harteille. [Watt, 2000]

Avainkuva määrittelee animoitavan objektin ominaisuudet yhdessä ajan hetkessä. Ominaisuudet ovat numeerisia muuttujia, jotka toistuvat jokaisessa kuvassa. Ominaisuus voi olla esimerkiksi objektin paikka, koko, asento, väri tai tekstuuri. Ominaisuus voi olla myös esimerkiksi valonlähde, kameran paikka tai sen syvyystarkkuus.

Ominaisuuksien muutos avainkuvien välillä lasketaan interpoloimalla välikuviin ominaisuuksille uudet arvot. Interpoloimalla tarpeeksi välikuvia saavutetaan jatkuva liike avainkuvien välillä. Interpolointi suoritetaan vain avainkuvien väliselle ajalle, jolloin parametrin arvoa on helppo ohjata haluttuun suuntaan.

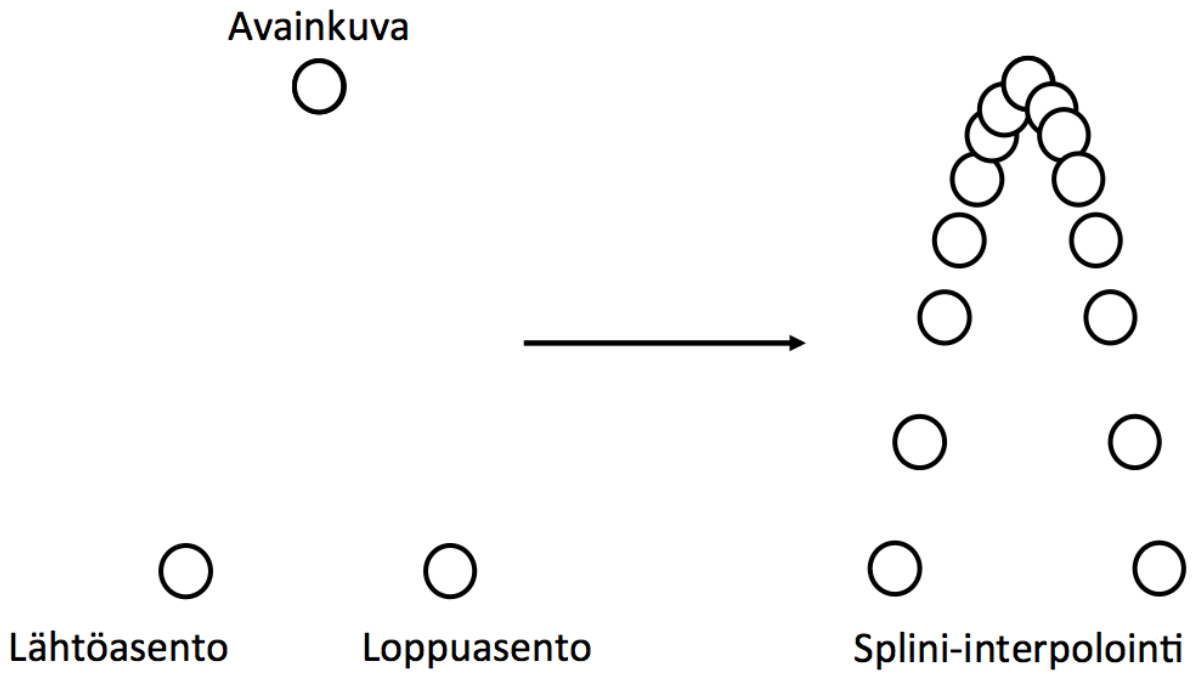
Interpoloinnin yksinkertaisin muoto on lineaarinen interpolointi. Siinä kohteen uusi paikka interpoloidaan suoran suhteen. Tällainen interpolointi toimii kuitenkin vain rajatuissa tapauksissa. Ajatellaan esimerkiksi yksinkertaista pomppivaa palloa. Avainkuvissa pallo on ylhäällä lakipisteessään ja alhaalla pinnalla, josta se pomppaa takaisin ylös. Lineaarisesti interpoloimalla pallon liikerata on suora laskeutuminen tason

pintaan ja siitä suora nousu lakipisteeseensä. Kuvassa 1 jokaisen kuvan välissä kuluu vakioaika, eli pallo nousee tasaisesi ylös ja samaa vauhtia alas.



Kuva 1. Pomppaavan pallon avainkuvat ja lineaarisella funktiolla interpoloidut välikuvat.

Kuten kuvasta 1 nähdään, lineaarisesti interpoloimalla saadaan hyvin jäykkä ja kulmikas liike, joka ei muistuta paljoakaan oikean pallon pomppaamista. Yksi vaihtoehto korjata liikettä olisi luoda uusia avainkuvia lähelle pallon lakipistettä, mutta avainkuvien määrä kasvaa nopeasti moninkertaiseksi. Helpompi tapa on kasvattaa interpolointifunktion astetta, jolloin liikkeestä saadaan luonnollisemman näköistä. Kuvassa 2 interpolointi on suoritettu toisen asteen funktiolla. Liikerata muistuttaa sitä, minkä ihmismieli mieltää pomppaavan pallon liikeradaksi.



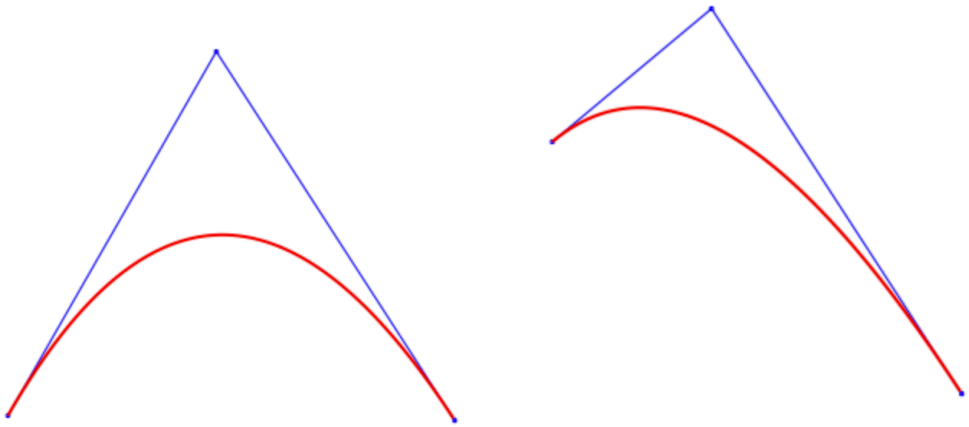
Kuva 2. Pomppaavan pallon avainkuvat ja toisen asteen funktiolla interpoloidut välikuvat.

Interpoloinneissa käytetään yleensä matala-asteisia polynomifunktioita niiden helpon laskettavuuden takia. Samalla ne ovat jatkuvia, mikä mahdollistaa sulavamman muutoksen animaatiossa, kun erillisiä kuvasarjoja liitetään yhteen pidemmäksi animaatioksi. Yksittäisellä polynomifunktiolla voidaan harvoin kuvata pidempää jatkuvaa muutosta. Siksi interpoloinnissa käytetään yleensä splinifunktioita, joiden jatkuvuus sovitetaan funktion määrittelyvälien päätepisteissä.

Yksi splinifunktio on Bezier-käyrä, jonka n:n asteen yleinen esitysmuoto on

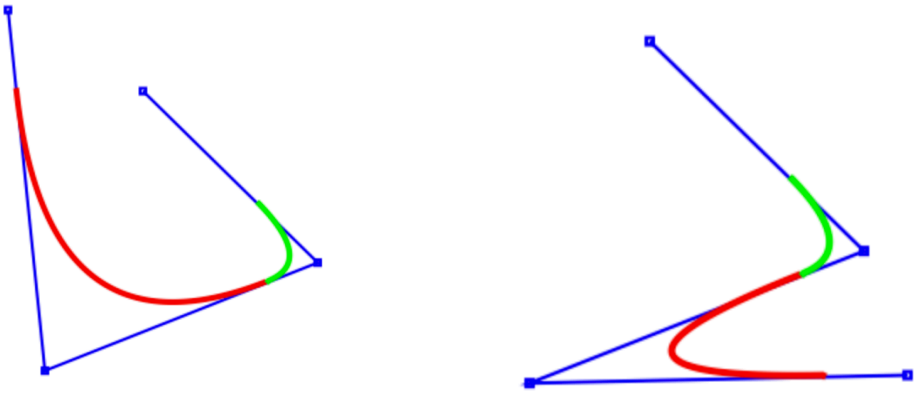
$$P(t) = \sum_{i=0}^{n} p_i \, B_{n,i}(t)$$ [Zhang and Liang, 2006].

Yhdellä Bezier-polynomilla voidaan kuvata koko käyrä. n-asteisella käyrällä on aina n+1 ohjauspistettä. Polynomille on tyypillistä se, että yhden ohjauspisteen muutos vaikuttaa aina koko käyrään. Kuvassa 3 on esitetty 2-asteisen polynomifunktion kuvaaja ja sen 3 ohjauspistettä. Kuvassa on liikutettu vasemmanpuoleista pistettä, jonka siirto vaikuttaa myös kuvaajan oikeaan reunaan.



Kuva 3. Yhden ohjauspisteen muutoksen vaikutus koko käyrään.

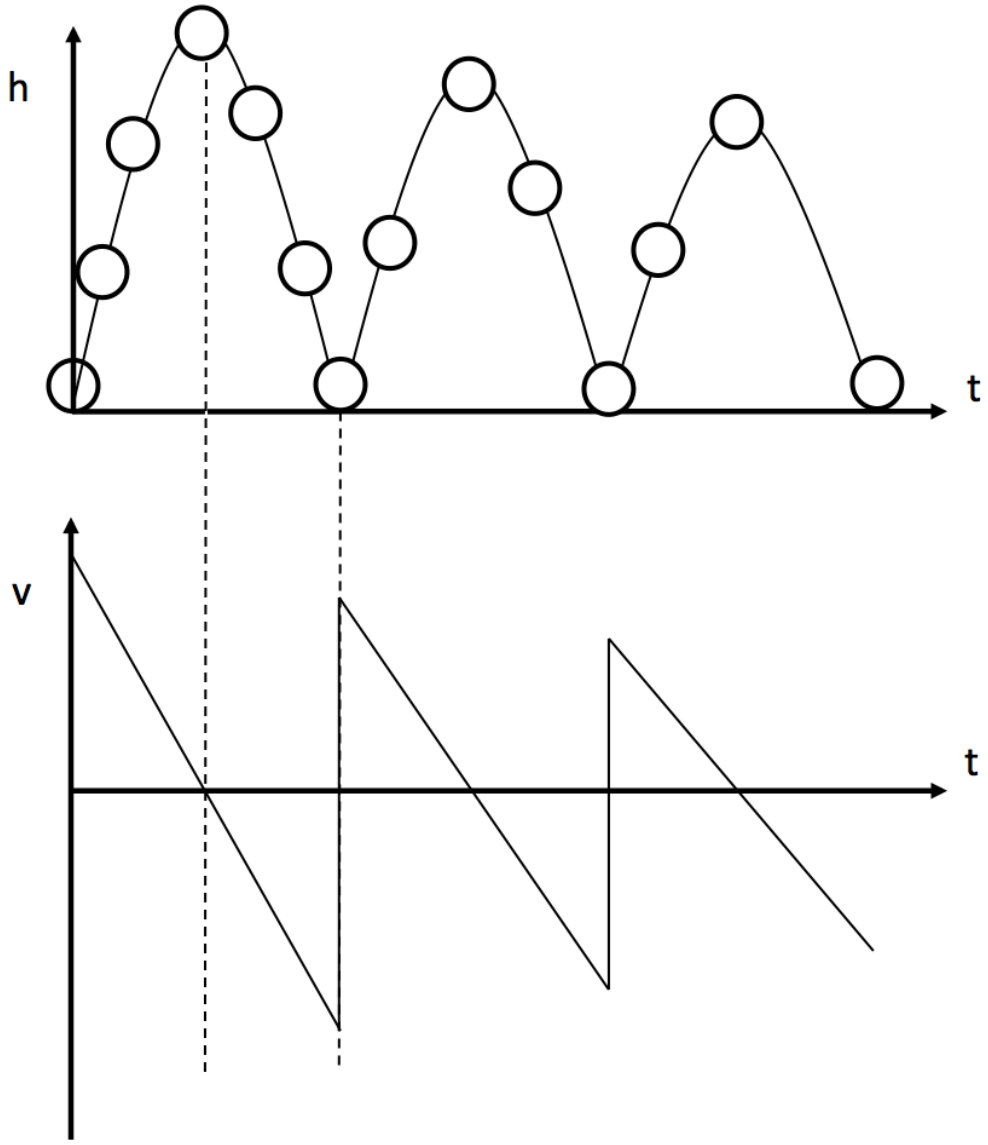
Toinen yleisesti käytetty funktio on B-splini-käyrä, joka on funktio, joka koostuu useammasta funktiosta. Toisin kuin Bezier-käyrä, B-splinin ohjauspisteen muutoksen vaikutus riippuu funktion asteluvusta. Korkeampiasteisilla funktiolla muutos on lokaali. Tämä on esitetty kuvassa 4.



Kuva 4. B-splinin ohjauspisteen muutos on lokaali.

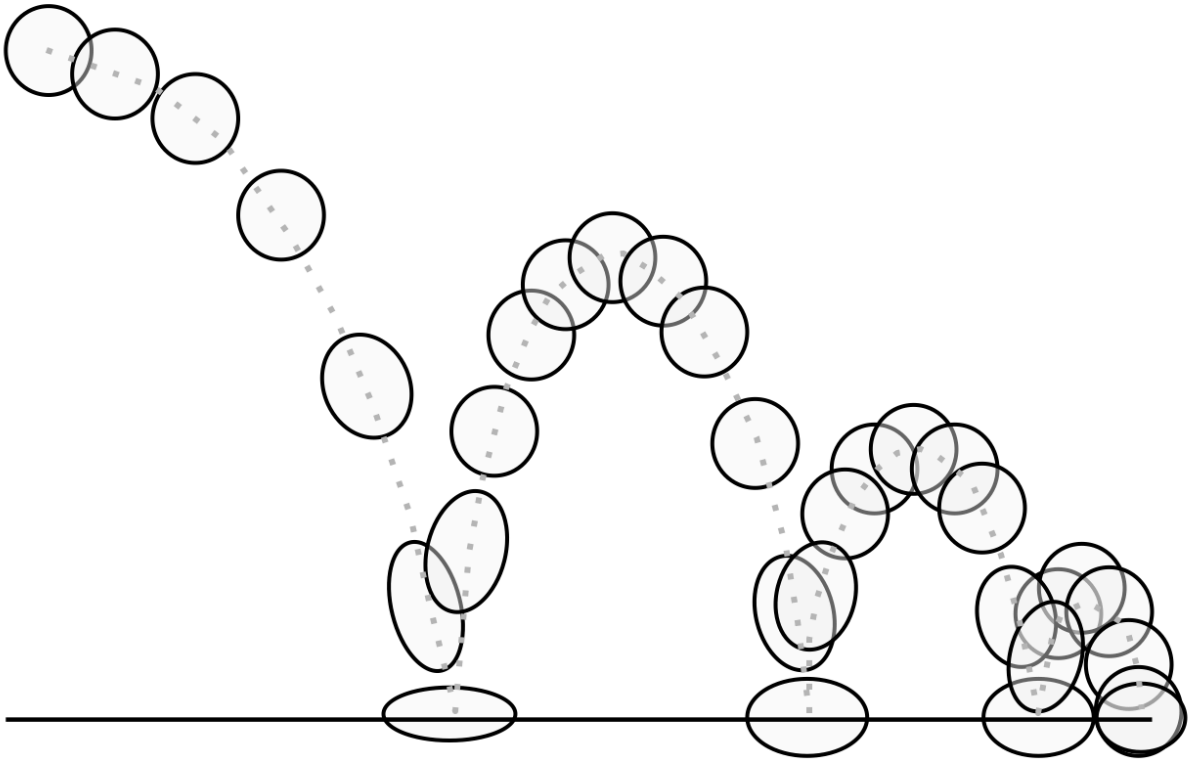
Interpoloinnin tuottama liike saattaa lähes lineaarisen interpoloinnin tuloksena näyttää tylsältä. Jos muistellaan vielä aikaisemmin mainittua pomppaavan pallon esimerkkiä, pallon liike ei todellisuudessa ole sama lähellä lakipistettään ja tasoa. Yleensä myös interpoloinnin yhteydessä tavoitellaan kiihtyvyyttä liikkeen alkuun ja jarrutusta liikkeen lopussa. Tämä voidaan saavuttaa yksinkertaisimmillaan käyttämällä useampaa

avainkuvaa liikkeen alussa ja lopussa. Aiemmin esitetyn kuvan 2 esimerkissä näkyy toisen asteen funktiolla interpoloitu liikerata, jossa avainkuvia on tiheämmässä liikeradan yläosassa. Jos avainkuvien välinen aika on vakio, niin useampi avainkuva aiheuttaa luonnollisesti hitaamman liikkeen. Jos taas halutaan nopeuttaa liikettä, avainkuvia pitäisi vähentää. Tämä ei välttämättä ole mahdollista ilman, että hukataan liikkeelle tärkeitä avainkuvia tai että lisätään tarpeettomia avainkuvia muualle animaatioon ja pienennetään kuvien välistä aikaa. Järkevämpi tapa säädellä liikkeen nopeutta on käyttää erillistä nopeusfunktiota. Nopeusfunktion avulla voidaan säätää avainkuvien välillä kulunutta aikaa jopa epälineaarisesti. Kuvassa 5 on esitetty saman pomppivan pallon esimerkin liikerata ja nopeusfunktio. Pallon noustessa kohti lakipistettä nopeus hidastuu ja kääntyy negatiiviseksi, kunnes törmäyshetken jälkeen nopeus vaihtuukin äkisti lokaaliin maksimiinsa.



Kuva 5. Pomppivan pallon nopeusfunktio.

Avainkuva-animoinnissa käytetään myös hyväksi Disney-studioitten aikanaan listaamia animoinnin perusperiaatteita [Lasseter, 1987]. Kuvassa 6 on lisätty aiemmin esitetyn pomppivan pallon esimerkkiin litistystä ja venytystä hetkeen, jolloin pallo osuu pintaan. Kuva vastaa enemmän sitä, jota ihmissilmä odottaakin näkevän katsellessaan pomppivaa palloa.



Kuva 6. Esimerkki "litistä ja venytä" -periaatteesta avainkuva animoinnissa.

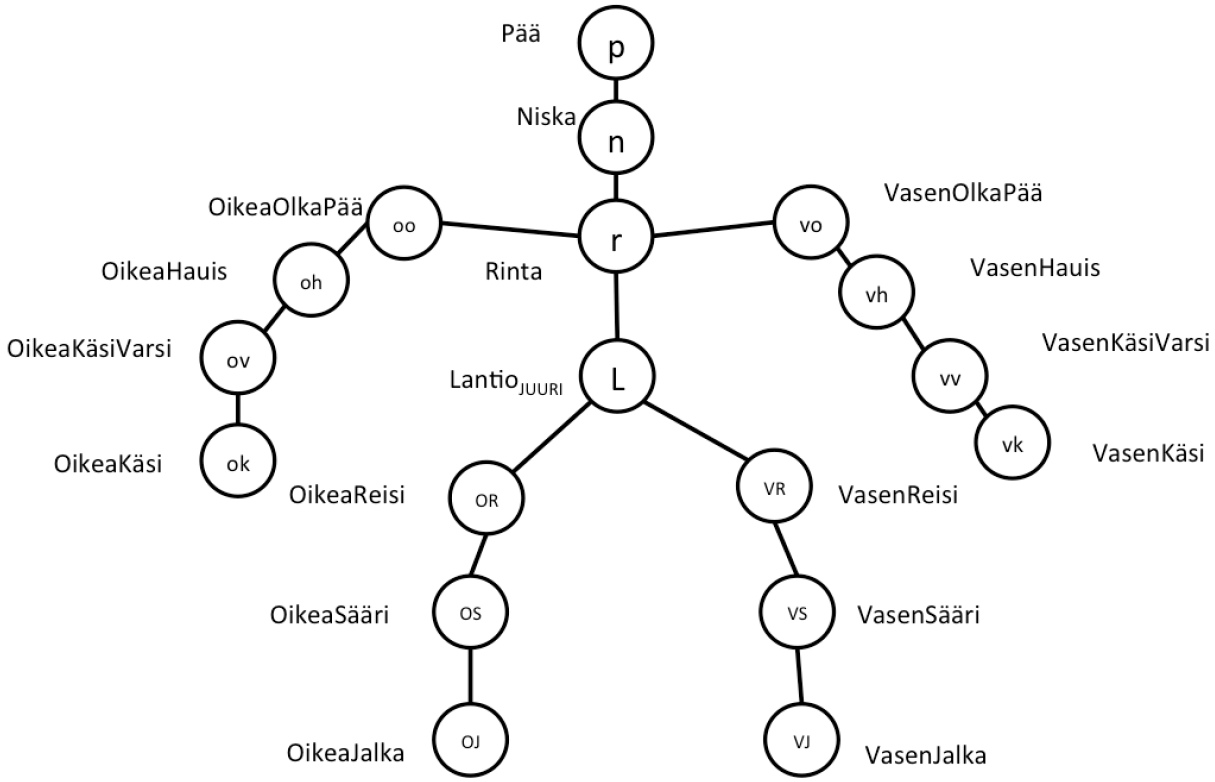
Kuten aiemmin esitetystä kuvasta 5 nähtiin, pomppivan pallon nopeus on suurimmillaan juuri ennen osumistaan pintaan tai siitä irrotessaan. Muokkaamalla pallon muotoa kapeammaksi lähellä pintaa saadaan aikaan illuusio vauhdin kiihtymisestä. Toisaalta juuri kun pallo osuu pintaan, sitä voidaan litistää, jotta iskeytymisen energia käy katsojalle selväksi.

Avainkuva-animoinnissa animoijalla on valta vaikuttaa suoraan animoitavan kappaleen parametreihin ja vapausasteisiin sen fysikaalisista ominaisuuksista riippumatta. Animaattori määrittelee kappaleen asennot tiettyinä ajanhetkinä, jonka jälkeen näiden hetkien välinen liike lasketaan interpololoimalla. Metodi on helppo käyttää ja se on suhteellisen yksinkertainen. Toisaalta metodin yksinkertaisuus asettaa sille rajoitteita. Kohde, jolla on useampia vapausasteita, on jo monimutkaisempi animoida. Tämän vuoksi esimerkiksi realistisesti liikkuvan ihmisen animointi tällä metodilla on lähes mahdotonta. Avainkuva-animointi onkin käytössä lähinnä yksinkertaisemmissa animaatioissa, joissa realistinen liike ei ole pääasia. Vaikkakin metodi on rajoitettu, se lienee eniten käytetty liikkeenhallintamekanismi animaatioissa.

Avainkuva-animaatioon liittyvä tutkimus keskittyy nykyään lähinnä siihen, kuinka helposti ja intuitiivisesti animaatioita tällä tekniikalla voidaan luoda. Esimerkinnä mainittakoon Igarashin ja kumppaneiden tutkimus [Igarashi et al., 2005], joka esittelee mekanismin, jolla animaattori voi luoda reaaliaikaisesti animaatiosekvenssin ohjaamalla hahmoa esimerkiksi hiirellä. Itse avainkuvien väliset kuvat luodaan tässäkin tapauksessa interpoloimalla.

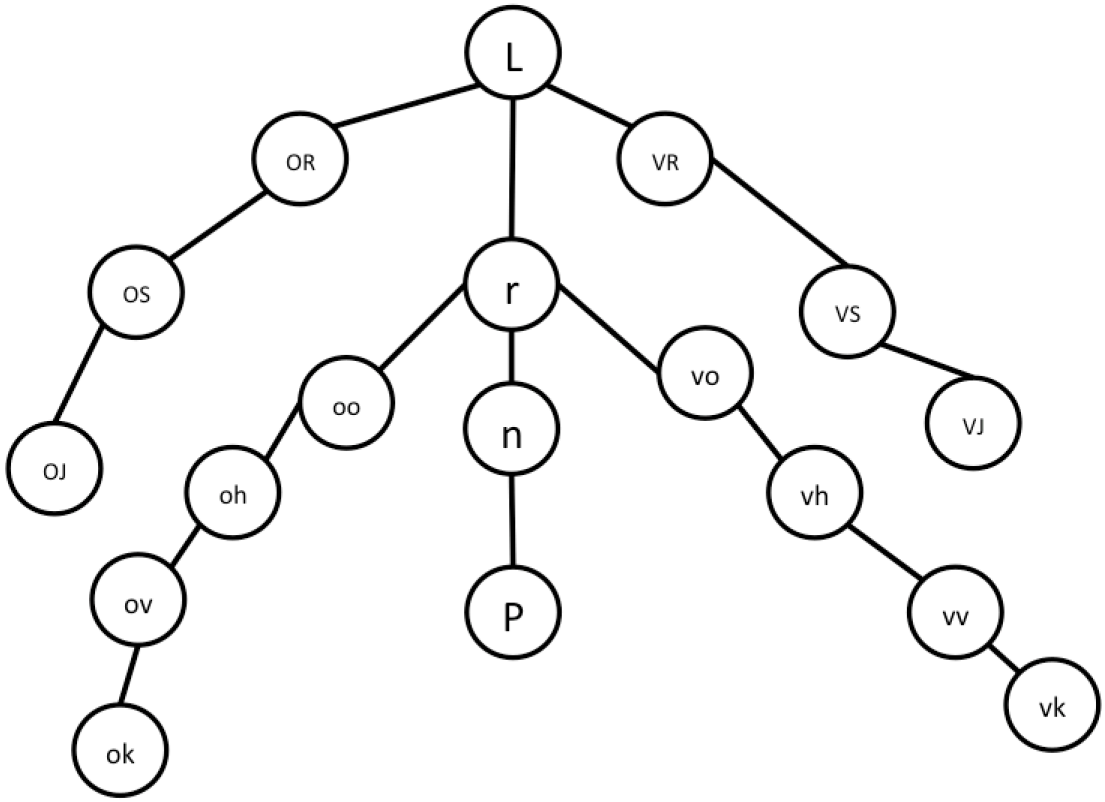
4 Hierarkkinen malli

Kuvattaessa kappaleen liikettä se voidaan kuvata joko globaalissa koordinaatistossa tai suhteessa toiseen kappaleeseen. Useimmiten on helpompaa ja järkevämpää kuvata liikettä nimenomaan jälkimmäisellä tavalla. Esimerkiksi kuun liike suhteessa maahan ja maan liike suhteessa aurinkoon ovat huomattavasti helpompia kuvata kuin kuun liike aurinkokunnan yhteisessä koordinaatistossa. Samanlaisia toisiinsa linkitettyjä liikkeitä löytyy myös kuvattaessa esimerkiksi ihmisen tai muun nivelikkään olion liikkeitä. Tällaiset toisiinsa sidoksissa olevat liikkeet ovat usein myös fyysisesti sidoksissa toisiinsa ja jollain tavalla rajoitettuja. Sidoksellisia ja rajoitettuja liikkeitä voidaan mallintaa hierarkkisella puurakenteella. Etenkin ihmisiä tai ihmisen kaltaisia nivelikkäitä olentoja animoitaessa puurakenne on hyödyllinen, koska raajojen kiinnittyminen on kuvattu rakenteeseen itseensä, jolloin animaattorin ei tarvitse välittää siitä. Kuvassa 7 on esitetty esimerkki ihmisen mallista. [Parent, 2008]



Kuva 7. Esimerkki ihmisen hierarkkisesta mallista.

Puun juurisolmu sisältää tiedon objektin paikasta globaalissa koordinaatistossa ja muiden solmujen sijainti ilmoitetaan aina suhteessa tähän solmuun. Jokainen kaari sisältää tiedon, joka kertoo kaaren päätepisteiden välisten solmujen sijaintitiedon suhteessa toisiinsa. Jokainen kaari päättyy solmuun, joka sisältää tarvittavan tiedon nivelten välisten linkkien muutokseen. Tieto voi olla esimerkiksi matriisi transformaatiota varten tai vaikka pinnan heijastavuuteen liittyvä ominaisuus. Juurisolmuun kohdistuva muutos vaikuttaa koko puuhun, samoin kuin johonkin sisäsolmuun kohdistuva muutos vaikuttaa kaikkiin sen lapsiin. Vastoin kuin luontevalta voisi tuntua, puun solmujen väliset kaaret kuvaavat nimenomaan niveliä ja solmut itsessään linkkiä nivelten välissä. Puun lehtisolmut kuvaavat mallin loppuvaikuttajia (end-effector), eli esimerkiksi animoitavan ihmisen kämmeniä. Kuvassa 8 on edellisessä kuvassa ollut hierarkkinen malli muunnettuna tutumpaan puu muotoon. [Parent, 2008]



Kuva 8. Ihmisen raajat puurakenteena.

Nivelikkään olion nivelillä on usein rajoituksia siinä, mihin suuntaan ja miten nivelet liikkuvat. Nivelen kykyä liikkua eri suuntiin kuvataan vapausasteilla (DOF - Degree of

Freedom). Kolmiulotteisessa koordinaatistossa vapaasti liikkuvalla nivelellä on yksi vapausaste jokaista akselia kohden.

Pelkän kohteen lisäksi on mahdollista mallintaa koko maailma hierarkkiseen puuhun. Tällöin maailman yksi kohde, jota halutaan animoida, koostuu puun haarasta. Transformaatiota varten juurisolmuksi valitaan kohteen alipuun ylin solmu, jonka alipuuhun transformaatio periytyy. Jäykkien liikkeiden lisäksi kappaletta voidaan muuntaa myös monipuolisemmin esimerkiksi venyttämällä, mutta tämä luonnollisesti riippuu mallinnettavasta kappaleesta. [Parent, 2008]

Luomalla hierarkkinen malli mahdollisista liikkuvista kappaleista ja niiden liikkeiden vaikutuksista toisiinsa autetaan animaattoria keskittymään vain oleellisiin parametreihin.

5 Kinematiikka

Vaikka avainkuva-animoinnissa animaattorilla on suuri kontrolli liikkeen lopputulokseen, sen voidaan sanoa olevan työläs tapa tuottaa liikettä. Kinemaattisilla järjestelmillä muodostettu liike luottaakin enemmän tietokoneen laskentatehoon ja kykyyn mallintaa saadusta datasta luonnollisen näköistä liikettä. Kinematiikassa liike muodostetaan liikkuvien kappaleiden välisillä suhteilla. Kuvaamalla kappaleen liike suhteessa edelliseen kappaleeseen saadaan muodostettua edellisessä luvussa kuvatun kaltainen puu. Puun solmut ovat yleensä kappaleita, jotka ovat fyysisesti sidoksissa toisiinsa. Fyysinen sidos on nivel kappaleiden välissä, jota puun kaaret kuvaavat. Puuesitysmuodolle on myös yleistä, että kappaleiden liike on rajoitettu. Rajoitus voi koskea vain nivelten kulmien asteita, mutta myös kappaleiden etäisyyksiä toisistaan. Kappaleiden ollessa kiinnittyneinä toisiinsa kappaleen siirtyminen vaikuttaa myös siihen sidoksissa oleviin kappaleisiin. Kinematiikassa kappaleen osat eivät muuta muotoaan, vaan liike rajoittuu nivelten vapausasteiden mukaan. Ajatellaan esimerkiksi puumallia ihmisestä ja tarkemmin mallin polviniveltä. Jotta mallin liikettä voidaan kuvata luonnollisesti, polven vapausasteita on järkevää rajoittaa yhteen sekä nivelen kulmaa on luonnollista rajoittaa vastaamaan normaalia polviniveltä. Polvi kääntyy vain yhdessä suunnassa, ja normaalitapauksessa se kääntyy vain taaksepäin. Lisäksi polven kiinnittämät sääri- ja reisiluu ovat normaali tapauksessa vakioetäisyydellä toisistaan.

Kinemaattiset menetelmät voidaan jakaa kahteen ryhmään, suoran kinematiikan menetelmiin ja käänteiskinemaattisiin menetelmiin. Suorassa kinematiikassa liikettä lähdetään muodostamaan alkuasennosta kohti haluttua asentoa. Käänteiskinematiikassa tutkitaan puolestaan sitä, millaisia liikeratoja pitää noudattaa, että tähän tilaan päädyttiin kun alkuasento on tiedossa.

5.1 Suora kinematiikka

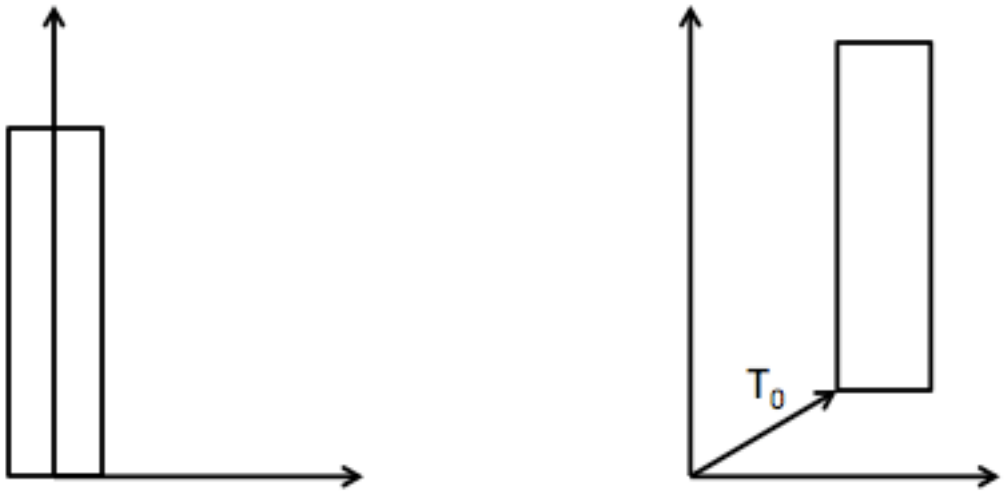
Suorassa kinematiikassa lähdetään alkuasennosta ja määritellään jokaisen liikkuvan nivelen kulma, jotta päästään haluttuun loppuasentoon. Animaattorin tehtävänä on ohjata mallin nivelten liikkeitä kohti haluttua asentoa. Suoran kinematiikan suuri etu on se, että se on laskennallisesti helppoa. Ongelmallista on taas se, että haluttua liikettä varten pitää kokeilla erilaisia variaatioita eri nivelille. Mitä monimutkaisempi kappale on, sitä enemmän niveliä ja sitä enemmän variaatioita samaan lopputulokseen johtavalle liikkeelle.

Suoralla kinematiikalla liikettä mallinnettaessa mallinnettavasta oliosta luodaan aiemmin kuvattu hiearkkinen puumalli. Animoitaessa valitaan haluttu juurikappaleen solmu juurisolmuksi, jonka mahdolliseen alipuuhun kaikki transformaatiot jäävät.

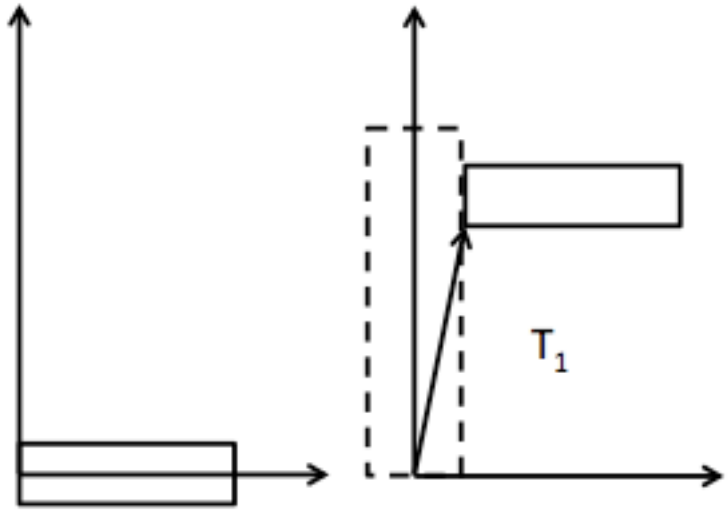
Juurikappaleen V_0 paikka globaalissa koordinaatistossa ($V´_0$) saadaan kertomalla $V´_0$ mahdollisella transformaatiolla T_0:

$$V´_0 = T_0 V_0 .$$

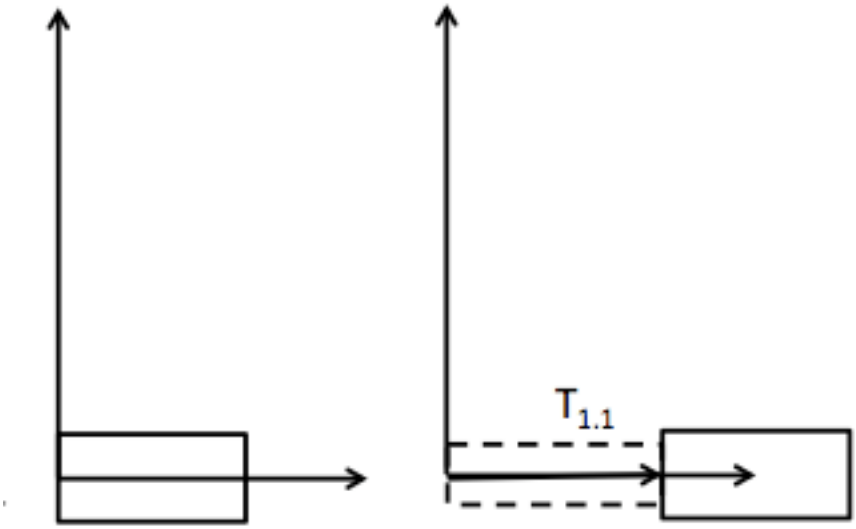
Juurisolmuunkin voi vaikuttaa transformaatio, joka tulee ottaa huomioon lopullista asentoa laskettaessa. Lisäksi malliin voidaan määritellä kiinteitä pisteitä, jotka eivät liiku animaation aikana. Esimerkiksi animoitaessa istuutuvaa ihmistä jalkaterä voidaan kiinnittää tasoon. Alipuiden juurikappaleeseen sidoksissa olevien linkkien translaatiot lasketaan itsenäisinä muutoksina suhteessa juurikappaleeseen ennen juurikappaleen translaatioita. Kuvissa 9-12 on kuvattu itsenäisten linkkien muutosten yhdistäminen.



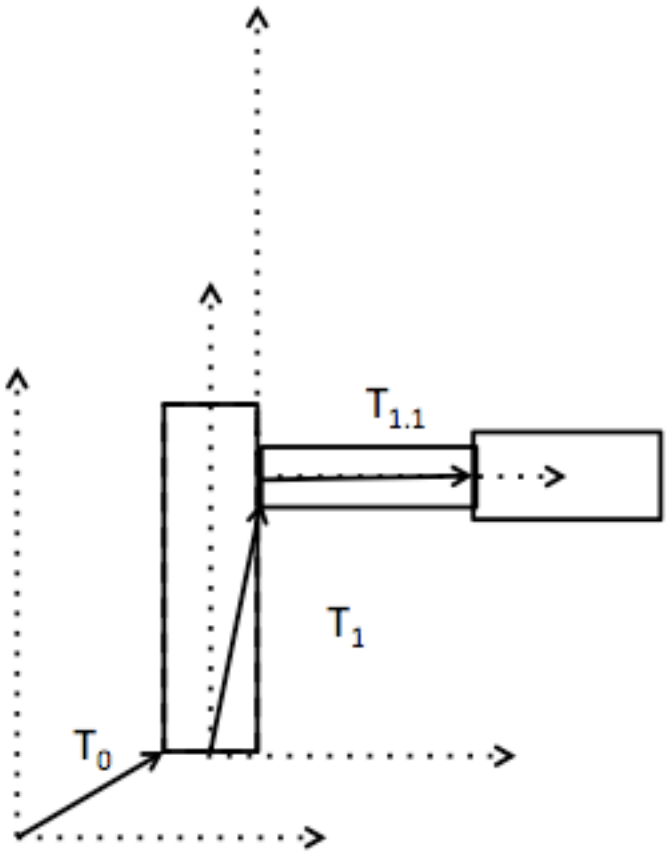
Kuva 9. Animoitavan mallin juurikappale ja sen translaatio.



Kuva 10.Linkin L_1 translaatio suhteessa muuttamattomaan juurikappaleeseen.



Kuva 11.Linkin $L_{1.1}$ translaatio suhteessa muuttamattomaan linkkiin L_1.



Kuva 12. Lopullinen asento kun eri linkkien translaatiot on yhdistetty.

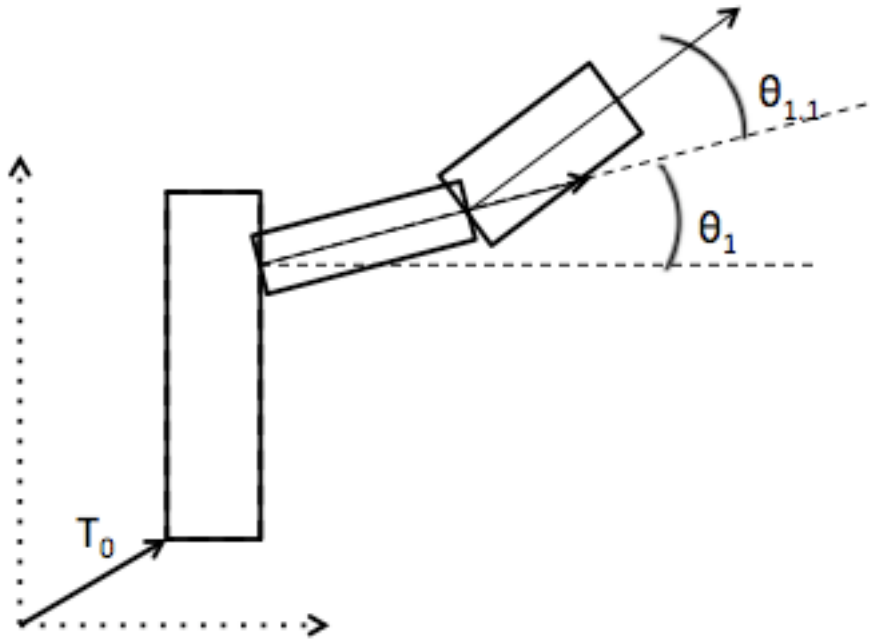
Samalla tavalla kuin aiemmin määriteltiin juurikappaleen paikka globaalissa koordinaatistossa, voidaan linkin $L_{1.1}$ paikka $V'_{1.1}$ määritellä:

$$V'_{1.1} = T_0 T_1 T_{1.1} V_{1.1} ,$$

jossa T_0 ja T_1 ovat vanhempiin kohdistuva translaatio ja $V_{1.1}$ linkin data, sisältäen ennen uutta transformaatiota tiedossa olevan vanhan transformaation. Jos niveleen kohdistuu translaation lisäksi rotaatio, mahdollinen rotaatio suoritetaan ennen translaatiota, mutta kuitenkin niin, että vanha transformaatio lisätään viimeisenä:

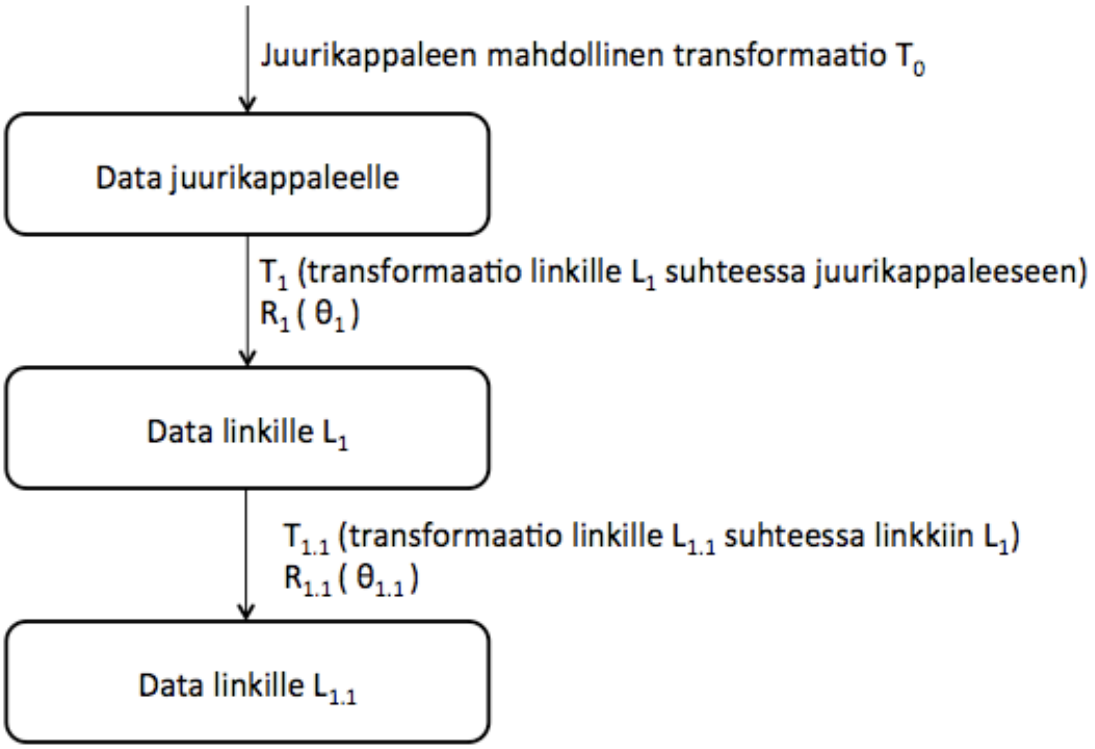
$$V'_{1.1} = T_0 \, T_1 \, R_1(\theta_1) \, T_{1.1} \, R_{1.1}(\theta_{1.1}) V_{1.1} .$$

Kuvassa 13 on esitetty rotaatio kuvien 9-12 translaatioihin yhdistettynä.



Kuva 13.Nivelten rotaatio yhdistettynä translaatioon.

Kullekin nivelelle lasketaan muunnos suhteessa juureen yhdistämällä niveleen kohdistuva muutos vanhempien muunnokseen. Jos juurisolmusta lähtee useampi alipuu, puuta seurataan syvyyshaulla. Jokaisen solmun transformaatio lisätään vanhemman transformaatioon, jolloin puun lehteen vaikuttaa koko polku juurisolmusta alkaen. Lopputuloksena on loppuvaikuttajan transformaatio katenoituna juuresta lehtisolmuun kuten kuvassa 14. [Parent, 2008]



Kuva 14.Transformaatio katenoidaan juuresta alkaen yhdeksi trasformaatioksi.

Monimutkaisemman liikkeen simuloiminen suoralla kinematiikalla saattaa olla työlästä, koska visuaalisesti tyydyttävän liikkeen aikaan saaminen on usein haettava kokeilemalla [Watt, 2000]. Esimerkiksi mallin jalan saaminen lattiaan saattaa alkutilasta riippuen vaatia transformaation lonkkanivelestä alkaen, kun taas seuraavaksi esiteltävän käänteiskinematiikan avulla jalka voidaan kiinnittää paikalleen heti simuloinnin alussa.

Suoraa kinematiikkaa käsitteleviä tutkimuksia ei oikeastaan löydy viime vuosilta, mutta kuten avainkuva-animointi, myös suoraa kinematiikkaa käytetään täydentämään jäljempänä esiteltäviä tekniikoita.

5.2 Käänteiskinematiikka

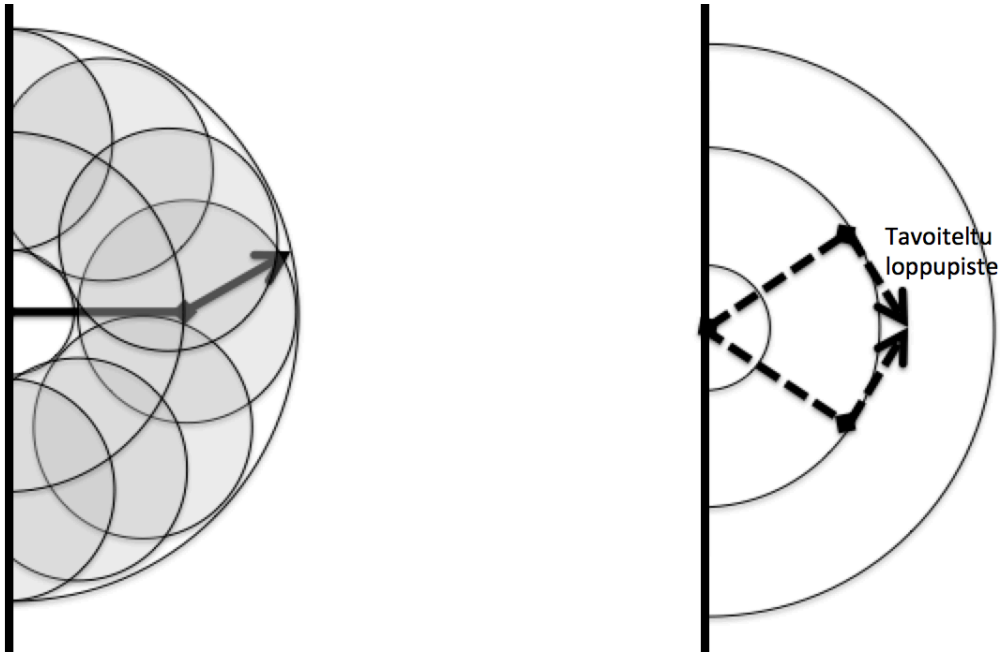
Käänteiskinematiikkaa (Inverse Kinematics) käytetään paljon myös robotiikassa. Teollisten robottien ohjaus perustuu pitkälti siihen, että robotti saavuttaa liikkeen lopuksi jonkun tietyn asennon, johon päästään liikuttamalla niveliä niiden vapausasteiden asettamissa rajoissa. Tiedossa on robotin lepoasento sekä asento, johon pyritään, jotta

haluttu toiminto voidaan suorittaa. Niin teollisuusroboteilla kuin ihmisilläkin on rajoitteita nivelten ja niiden liikkumisen suhteen.

Käänteiskinematiikkaa käytettäessä animoijan tehtäväksi jää määritellä alkuasento ja loppuvaikuttajalle koordinaatit liikkeen loputtua. Ohjelmallisesti lasketaan kullekin nivelelle tarvittava kulman muutos. Käänteisen kinematiikan etuna suoraan kinematiikkaan nähden voidaan pitää sitä, että animaattorin on helpompi saavuttaa haluttu loppuasento ja paikka animoitavalle kohteelle. Kun suorassa kinematiikassa haluttu loppuasento löydetään yrityksen ja erehdyksen kautta, käänteiskinematiikassa loppuasento on asetettu alkuparametri. Käänteiskinemaattisen järjestelmän voidaan katsoa olevan musta laatikko, joka saa syötteinään loppuvaikuttajan alku- ja loppukoordinaatit. Se, että loppuvaikuttajan paikka voidaan määritellä heti aluksi, auttaa myös välttämään interpoloinnin tuomat epätarkkuusongelmat, esimerkiksi jalan uppoamisen pinnan sisälle.

Kohdetta animoitaessa kohteesta luodaan nivelletty malli, jonka nivelillä on haluttu määrä vapausasteita. Malli voidaan tallentaa aiemmin kuvattuun hierarkkiseen puurakenteeseen, jolloin juureen kohdistettu muutos vaikuttaa myös loppuvaikuttajassa. Algoritmien tehtävä on laskea kullekin nivelelle vaadittava kulma, jotta saadaan aikaan liike alkutilasta lopputilaan. Algoritmin hyvyydestä riippuen liike voi olla luonnollinen tai luonnoton. Liike etenee inkrementaalisesti, kunnes päästään haluttuun asentoon tai käyttäjän määrittämälle etäisyydelle siitä. Mitä monimutkaisempi liike on kyseessä, sitä vaikeampaa on laskea halutut kulmat eri nivelille. Käänteiskinematiikka toimii hyvin robotiikassa sen ei-realistisen luonteen ja nivelten rajoitusten vuoksi, mutta tietokoneanimaatiossa tuotettu liike ei välttämättä ole aina kaikkein luontevin.

Yksinkertainen käänteiskinemaattinen ongelma voidaan ratkaista analysoimalla nivelvarren geometriaa ja koordinaatistoa, johon sillä on mahdollisuus ulottua. Esimerkiksi kuvassa 15 on yksinkertainen nivelletty varsi, joka on nivelletty kiinni seinään. Molemmissa nivelissä on yksi vapausaste. Analysoimalla varsien pituuksia saadaan laskettua avaruus, johon loppuvaikuttajalla on mahdollisuus ylettyä. Tämän avaruuden jokaisella pisteellä on korkeintaan kaksi mahdollista arvoa nivelten kulmiksi.



Kuva 15. Analysoimalla rakennetta voidaan päättelemällä ratkaista käänteiskinemaattinen ongelma.

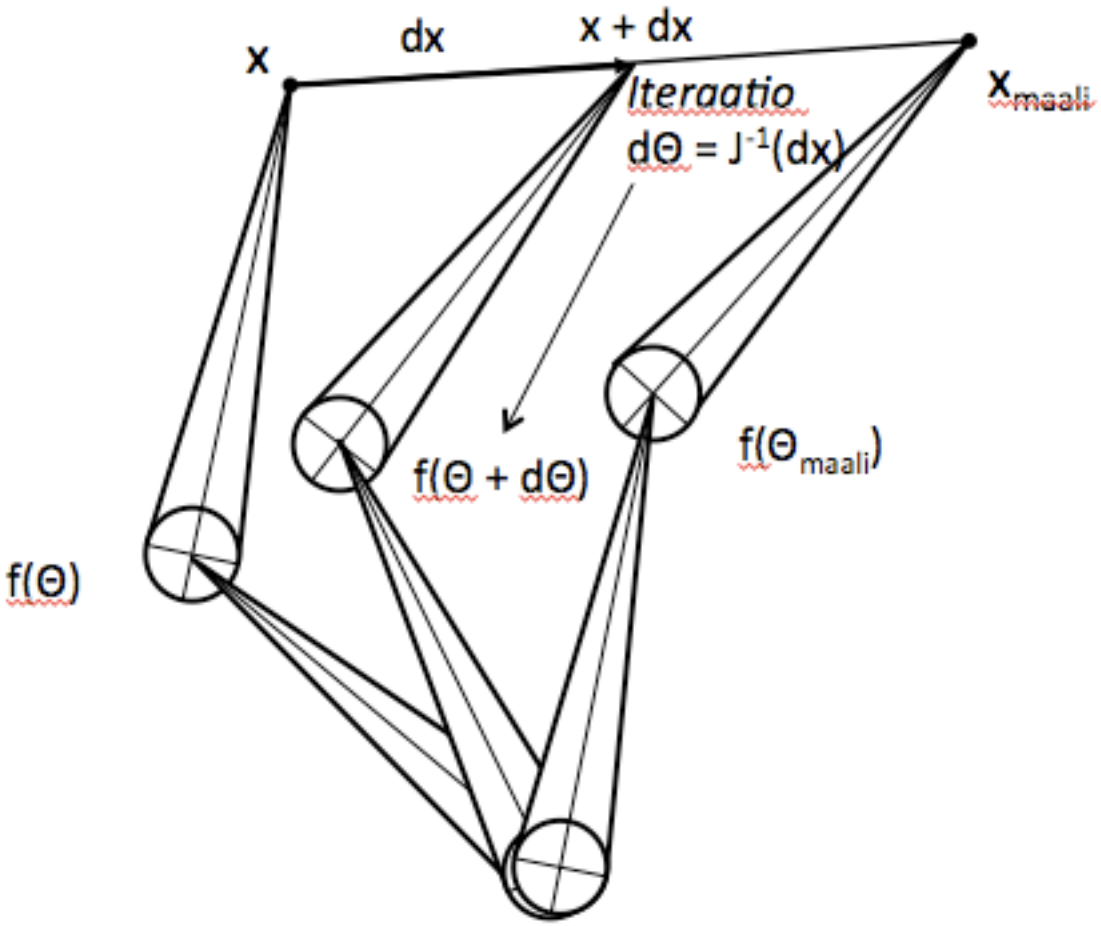
Vähinkin monimutkaisemmissa tapauksissa voidaan käyttää Jacobin matriisia. Jacobin matriisilla kuvataan lineaari- ja kulmanopeusvektorit, joilla päästään nykyisestä asennosta haluttuun loppuasentoon [Buss, 2009].

Jos Jacobin matriisi on neliömatriisi, saadaan kulmanopeudet laskettua lineaarisesti sen käänteismatriisin avulla:

$$J(\theta) = \frac{dx}{d\theta}$$

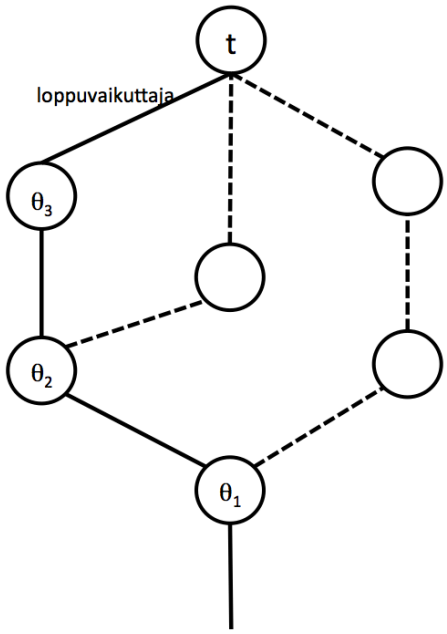
$$J^{-1}(\theta)(dx) = d\theta\,.$$

Ongelma ratkaistaan siirtämällä loppuvaikuttajaa pienin muutoksin kohti haluttua pistettä. Yksi tällainen iteraatio on kuvattu kuvassa 16.



Kuva 16. Yksi iteraatio kohti maalia [Watt, 2000].

Jos käänteismatriisia ei löydy, voidaan käyttää Jacobin matriisin pseudokäänteismatriisia. Tällainen tilanne syntyy esimerkiksi silloin, kun nivelvarrella on useampia mahdollisuuksia päästä loppuratkaisuun, eikä ongelmalle ole yksittäistä ratkaisua. Esimerkiksi kuvassa 17 nivelellä on useita mahdollisuuksia saada loppuvaikuttaja pisteeseen t. Kuvaan on piirretty kolme eri mahdollisuutta.



Kuva 17.Esimerkki käänteismatriisittomasta konfiguraatiosta.

Yllä mainittu tilanne saadaan esimerkiksi, kun simuloidaan ihmisen käsivartta. Useamman ratkaisun tapauksessa simuloitua liikettä voidaan viedä haluttuun suuntaan lisäämällä esimerkiksi rajoituksia. Kallmann [Kallmann, 2008] käyttää nimenomaan rajoituksia, joilla päästään hyvään ratkaisuun.

Toinen tapaus käänteismatriisittomasta konfiguraatiosta on tilanne, jossa esimerkiksi kaksinivelinen varsi on lähtöasennossa ojennettuna suoraksi ja loppuvaikuttajan maali on 180 astetta toiseen suuntaan [Watt, 2000].

Käänteiskinematiikassa on käytetty myös vaihtoehtoista Jacobin matriisia, jossa Jacobin pseudokäänteismatriisissa käytetään loppuvaikuttajan koordinaattien sijaan kohdepisteen koordinaatteja. Kyseisen menetelmän tulokset ovat samansuuntaisia normaalin tavan kanssa [Parent, 2008].

Sen sijaan, että käytetään Jacobin matriisia, käänteiskinemaattista ongelmaa voidaan käsitellä optimointiongelmana. Jos esimerkiksi x(q) on loppuvaikuttajan nykyinen paikka ja p on haluttu maali, muuttamalla vektorin q arvoa loppuvaikuttaja siirtyy joko lähemmäs haluttua maalia tai kauemmas. Silloin esimerkin minimoitavan virheen E(q) yhtälö olisi:

$$E(q) = (p - x(q))^2 .$$

Pelkän etäisyyden sijaan minimointiyhtälö voi olla mikä tahansa muu vektorin q ominaisuus [WELL-1989]. Käytetty optimointimenetelmä riippuu virhefunktiosta. Optimointiongelmat ovat kokonaan oma tutkimusalueensa, eikä niitä käsitellä tässä enempää.

Cyclic Coordinate Descent -menetelmässä jokaista niveltä tarkastellaan erikseen uloimmasta sisimpään. Jokaiselle nivelelle valitaan kulma, joka vie loppuvaikuttajaa lähimmäksi kohdepistettä. Minimoitava virhe on loppuvaikuttajan paikka- ja kulmavektoreiden poikkeama maalista. Menetelmän eri variaatioissa nivelen kulman muutoksen hyvyyttä verrataan muihin niveliin ja eri painotuksia käyttäen paras muutos hyväksytään kyseisellä iteraatiolla [Parent, 2008]. Menetelmä toimii myös edellä mainitussa käänteismatriisittomassa tapauksessa, jossa loppuvaikuttajan maali oli samalla akselilla lähtöpisteen kanssa [WELL-1989].

Hecker ja muut [Hecker et al., 2008] kehittivät aikaisemmista poikkeavan tavan ratkaista käänteiskinemaattisia ongelmia. Kyseinen tapa kehitettiin tietokonepelissä olevia fantasiaolentojen evoluutiota varten, joiden fysiikka poikkeaa huomattavasti tunnettujen eliöiden fysiikasta. Kuvassa 18 on kuvankaappaus kyseisestä pelistä ja siinä käytetyistä hahmoista. Pelissä pelaaja saa itse määritellä otuksen fysiikkaa, joten pelin liikkeen mallinnusalgoritmien tulee soveltua hyvin erilaisille rakenteille.

Kuva 18.Kuvankaappaus Spore-tietokonepelistä (http://www.ea.com).

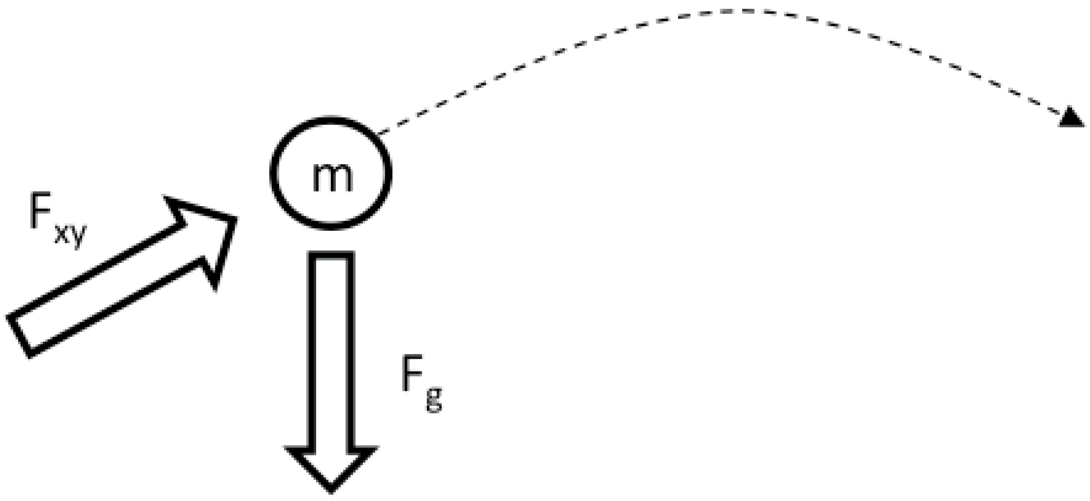
Vaikka käänteiskinematiikka tarjoaa ratkaisuja moniin suoran kinematiikan ongelmiin, ei sekään ole täysin ongelmatonta. Realistisen liikkeen aikaansaaminen vaatii paljon työtä animaattorilta.

5.3 Proseduraalinen metodi

Proseduraalinen metodi on tehokas tapa optimoida tietyntyyppisiä liikkeitä, kuten kävely ja juoksu. Yhteistä näille on toistuva liike. Animaattorin tehtävä on kuvata liikkeelle välttämättömät argumentit, esimerkiksi askeleen pituus tai nopeus, jonka jälkeen liike sovitetaan liikettä varten määritettyyn proseduuriin. Metodin ongelma on sen rajoittuneisuus vain tietyntyylisiin liikkeisiin. Esimerkiksi yleistä liikehdintää ei voi mallintaa proseduraaliseksi [Huang, 1996].

6 Dynamiikka

Siinä missä edellisten lukujen liikkeenhallintametodit keskittyivät loppuvaikuttajien lopulliseen paikkaan ja asentoon, dynaamiset metodit keskittyvät enemmän liikkeen laatuun kuin tarkkuuteen. Dynaamisessa liikkeenhallinnassa animoijan tulee tuntea animoitavan kappaleen fysikaaliset ominaisuudet. Ominaisuudet voivat vastata reaalimaailmaa, mutta voivat poiketa niistä myös huomattavastikin. Liikettä hallitaan määrittelemällä kappaleelle fysikaalisia parametreja, esimerkiksi massa ja kappaleen alkunopeus, sekä kappaleeseen vaikuttavat ulkoiset voimat. Yksinkertaisena esimerkkinä voidaan tarkastella kuvan 19 palloa, johon kohdistuu painovoiman F_g lisäksi toinen voima F_{xy}, joka antaa pallolle ylös ja sivulle vaikuttavaa liikettä.



Kuva 19.Pallon liikerata lasketaan massan ja vaikuttavien voimien perusteella.

Dynaamisissa järjestelmissä animaattorin ei tarvitse välittää kappaleen vapausasteista ja nivelistä. Kappaleen ominaisuuksien avulla määritellään se, kuinka kappale käyttäytyy, kun siihen kohdistetaan ulkoisia voimia. Animoija ei kontrolloi itse liikettä, ainoastaan kappaleen ominaisuuksia ja siihen kohdistettavia voimia. Tähän puutteeseen on kehitetty erilaisia optimointimenetelmiä rajoittamalla liikettä erilaisin rajoituksin [Huang, 1996].

Yksittäisiä kuvia luotaessa kappaleisiin vaikuttavat voimat otetaan huomioon ja kappaleille lasketaan uusi paikka ja kappaleisiin vaikuttavat voimat päivitetään.

Tyypillinen kappaleisiin vaikuttava voima on gravitaatiovoima, joka on helppo laskea kappaleiden massojen ja etäisyyksien avulla. Toinen usein käytetty voima on jousivoima. Jousen aiheuttama voima voi vaikuttaa kahteen eri suuntaan. Venytettäessä jousta lepopituuden ylitse se vetää kappaleita puoleensa, kun taas puristettaessa jousta kasaan se työntää kappaleita erilleen. Voiman aiheuttamaan muutoksen voidaan vaikuttaa myös muuttamalla ympäristön ominaisuuksia. Esimerkiksi muuttamalla kappaleiden välisen aineen viskositeettia voidaan vaikuttaa kappaleen nopeuteen. Suorien voimien lisäksi kappaleisiin voidaan kohdistaa vääntäviä voimia. Vääntävään voimaan liittyy voiman lisäksi kulma, johon vääntö kohdistuu.

Yksi usein käytetty elementti dynaamisissa metodeissa on jousi-massa-vaimennin-elementti. Sen avulla voidaan helposti kuvata joustavia objekteja ja pitää kappaleita tietyllä etäisyydellä toisistaan. Jousi-massa-vaimennin-systeemeitä käytetään avuksi simuloitaessa ulkopuolisten voimien vaikutuksia nivelikkäisiin olioihin. Mallinnettaessa oliota sen hierarkkisen mallin kaaret ovat jousia ja solmut massoja. Kun solmuun kohdistetaan ulkopuolinen voima, solmu liikkuu verrattuna olion muihin solmuihin. Jousien avulla solmuun kohdistunut voima siirtyy myös viereisiin solmuihin palaten sieltä jouselle tyypillisellä tavalla takaisin alkuperäiseen solmuun. Solmujen i ja j välillä olevan jousen aiheuttama voima voidaan laskea Hooken lain avulla:

$$F_{ij} = k_s\left(Lc_{ij}(t) - Lr_{ij}\right)v_{ij}\ .$$

Tässä k on jousivakio, Lc on kyseisten solmujen välimatka, Lr on jousen pituus levossa ja v on yksikkövektori solmujen i ja j välillä.

Solmuun kohdistetun voiman suuruudesta ja ajanjaksosta riippuen saatetaan saada aikaan liike, joka kasvaa koko ajan. Tätä ehkäisemään malliin lisätään vaimentimia, joilla hidastetaan voimaa, jolla jousi muuttaa nopeuttaan. Vaimentimen voima on aina päinvastainen jouseen muutosta aiheuttamaan voimaan. Sen avulla voidaan stabiloida malli rajoittamalla jousen pituutta tietylle välille.

Jos kappaleeseen ei vaikuta voimia, sen kiihtyvyys on nolla ja nopeus on vakio. Silloin pisteen paikkaan ajan t kuluttua vaikuttaa vain sen nopeus:

$$x(t + \Delta t) = x(t) + v(t) * \Delta t\ .$$

Kappaleen kiihtyvyys voidaan olettaa vakioiksi ajan kuluessa, jotta laskenta olisi yksinkertaisempaa. Tällä oletuksella kappaleen keskimääräinen nopeus ajan kuluessa on sen alku- ja loppunopeuksien keskiarvo. Tästä saadaan kaava jolla saadaan kiihtyvyydellä a kiihtyvän kappaleen sijainti ajan t kuluttua:

$$x(t + \Delta t) = x(t) + v(t) * \Delta t + \frac{1}{2} * a(t) * \Delta t^2 .$$

Kiihtyvyyden käsittely edellä kuvatulla tavalla saattaa useissa tapauksissa olla liian yksinkertaistettu. Esimerkiksi juuri nivellettyjen olioiden tapauksessa kiihtyvyys vaihtelee ajan suhteen ja usein vielä ei-lineaarisesti. Tarkkuutta voidaan parantaa helposti jo käyttämällä esimerkiksi Eulerin menetelmää, mutta parempaan tulokseen päästään käyttämällä Runge-Kutta-menetelmää. Nostamalla Runge-Kutta-menetelmän kertalukua päästään aina parempaan tarkkuuteen, mutta tarvittavien laskentaresurssien kustannuksella. Yleisesti käytetty kertaluku on neljä tai viisi. [Parent, 2008]

Suoran nopeuden ja kiihtyvyyden lisäksi kappale voi olla kiertävässä liikkeessä akselin ympäri. Jos mallinnetaan yksittäistä pistettä, kiertoliikkeellä ei ole merkitystä. Kiertoliikettä kuvataan kulmanopeusvektorilla, jonka suunta kertoo akselin suunnan ja koko kertoo kierrosten määrän akselin ympäri tietyssä ajassa. Huomattavaa kulmanopeudessa on se, että kappaleen kiertäessä n kertaa akselin ympäri sen kulmanopeus on sama, vaikka akseli sijaitsee kappaleen keskipisteessä tai kauempana kappaleesta. [Parent, 2008]

Suoraan kulkevan voiman kohdistaminen kappaleeseen aiheuttaa kappaleen kiihtyvyyden kasvua. Kiihtyvyys saadaan laskettua kappaleen massan avulla. Jotta saadaan selville, mihin suuntaan kappaleen kiihtyvyys kasvaa, pitää tietää massan jakautuminen kappaleessa ja sen suhde kohdassa, johon voima osuu. Kappaleen kokonaismassa on yksittäisten massojen summa. Kuten aiemmin mainittiin hierarkkisen mallin yhteydessä, simuloitavan objektin massa on yleensä sijoitettu objektin kaariin. Jos massan paikka koordinaatistossa ajan hetkellä t on $q_i(t)$, yksittäinen massa on m_i ja kokonaismassa on M, saadaan laskettua kappaleen massakeskipiste x(t) kaavalla

$$X(t) = \frac{\Sigma(m_i * q_i(t))}{M}.$$

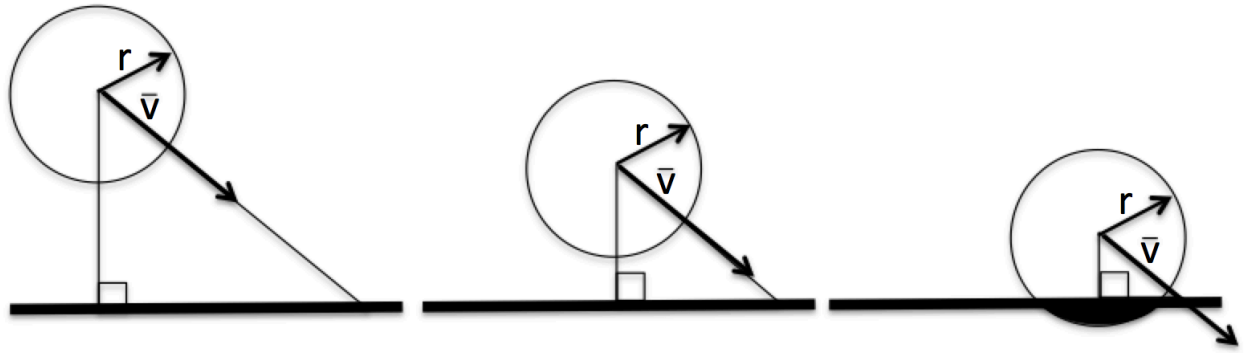
Riippuen voiman osumakohdasta suhteessa kappaleen massakeskipisteeseen voima voi lineaarisen kiihtyvyyden lisäksi aiheuttaa kappaleelle myös kiertoliikettä. [Parent, 2008]

Suljetuissa systeemeissä voiman säilymisen periaatetta kuvataan momentilla. Kappaleen momentti muodostuu sen liikkeestä ja kappaleen massasta. Momentti voi olla lineaarista tai vääntävää. Momentti sijoitetaan massakeskipisteeseen. Suljetussa järjestelmässä koko kappaleen momenttien summa säilyy, joten myös järjestelmän tuntemattomien osien järjestelmien momentit saadaan laskettua muutoksen jälkeen. Lineaarisen momentin ominaisuuksiin kuuluu, että se on suoraan verrannollinen pisteeseen vaikuttavan lineaarisen voiman kanssa. Suoran voiman vaikutus on suora muutos massakeskipisteessä olevaan momenttiin. Vääntömomentin ja vääntävän voiman välillä on samanlainen suhde. Jos kappaleeseen ei vaikuta vääntävää voimaa, vääntömomentti on vakio. Kuitenkin kappaleen kulmanopeus voi muuttua, vaikka kappaleeseen ei kohdistu vääntävää voimaa, esimerkiksi silloin kun kappaleen massan jakautuminen muuttuu. Massan siirtyessä lähemmäs akselia sen kulmanopeus kasvaa. Koska vääntömomentti on kulmanopeuden, massan ja massakeskipisteen sekä massan välisen etäisyyden funktio, kulmanopeuden tulee kasvaa, kun massakeskipisteen ja massan välinen etäisyys pienenee, jotta vääntömomentti pysyy samana. [Parent, 2008]

6.1 Törmäykset

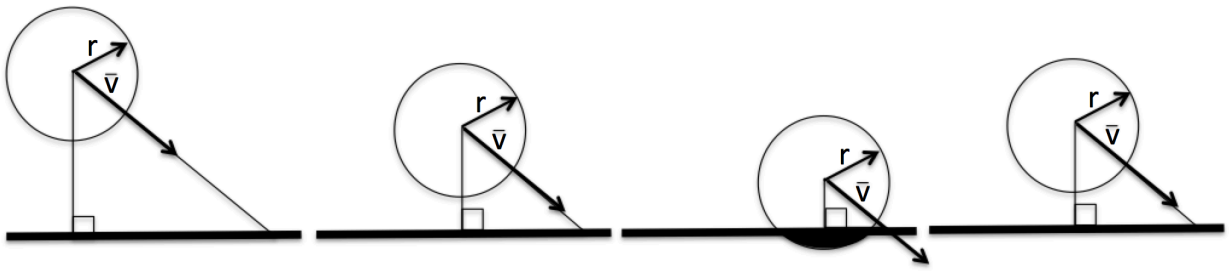
Kappaleiden liikkuessa vapaasti avaruudessa tulee huomioida niiden mahdolliset törmäykset toisiinsa tai staattisiin pintoihin. Tämän lisäksi etenkin dynaamisessa mallinnuksessa tulee miettiä sitä, mitä törmäyksestä seuraa. Tässä suhteessa kappaleiden geometria ei näyttele niin suurta osaa kuin kappaleiden massojen jakautuminen kappaleessa. Törmäyskohdasta riippuen vaikuttava voima voi olla suoraan vastakkaiseen suuntaan tai vääntävä. Törmäysten havainnoimiseksi kappaleiden liikeratojen mahdolliset päällekkäisyydet tutkitaan aina tietyllä ajanjaksolla.

Törmäysreaktion ajoittamiseen on kaksi metodia. Ensimmäinen tapa on jatkaa niin pitkään kuin mahdollista aina törmäyshetkeen saakka, jonka jälkeen törmäyksen vaikutus lasketaan. Tässä tapauksessa kappaleet uppoavat toisiinsa ennen kuin törmäyksen vaikutus nähdään. Tilanne on kuvattu kuvassa 20. Muun muassa kappaleen nopeuksista riippuen menetelmällä saadaan usein varsin siedettäviä tuloksia. Lisäksi menetelmä on helppo toteuttaa. [Parent, 2008]



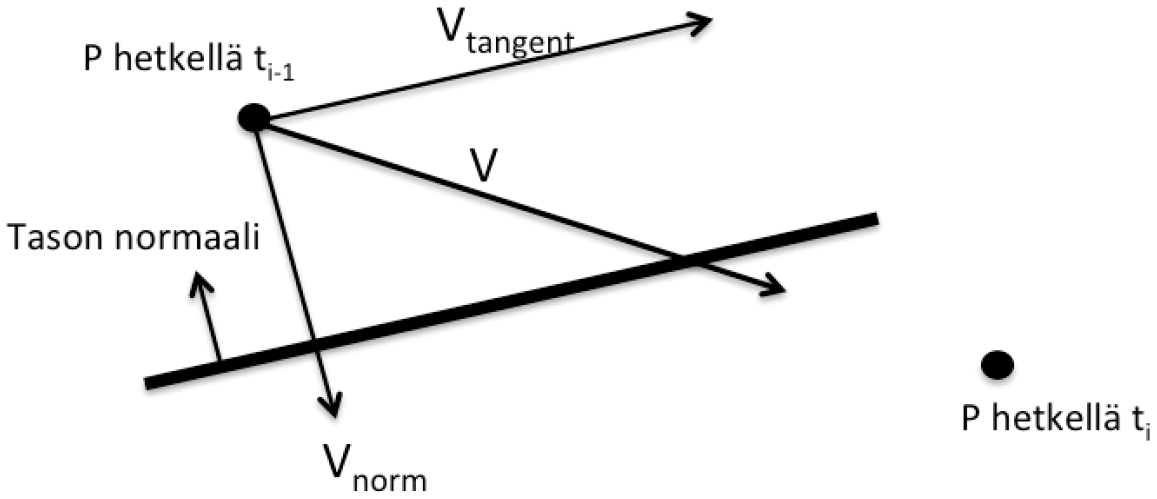
Kuva 20. Törmäyksen reaktio lasketaan hetkellä t_i, jolloin kappale on jo ehtinyt tunkeutua tasoon.

Toinen menetelmä on kuvattu kuvassa 21. Menetelmässä peruutetaan ajassa taaksepäin ja lasketaan törmäyksen vaikutus jo ennen itse törmäystä. Menetelmä on huomattavasti tarkempi kuin ensimmäinen tapa, mutta etenkin ympäristössä, jossa tapahtuu useita törmäyksiä, jatkuva peruuttaminen ja uudelleen laskenta voi käydä liian raskaaksi. Molempien menetelmien hyvyys riippuu myös kuvien välisestä ajanjaksosta ja siitä, miten kappale on sijoittunut suhteessa törmäys kohteeseen hetkellä t_{i-1}. [Parent, 2008]



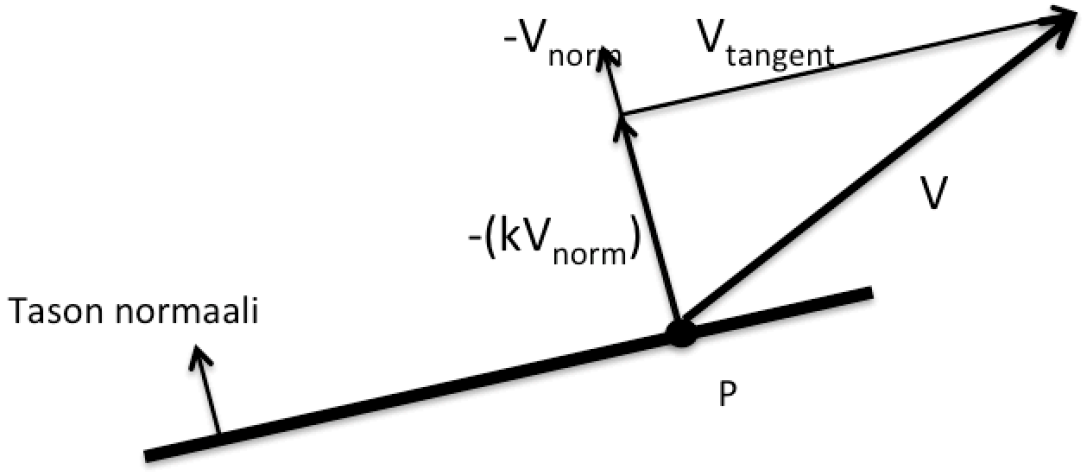
Kuva 21. Törmäyksen reaktio lasketaan hetkellä t_{i-1}, joka on tilanne juuri ennen törmäystä.

Törmäyksen vaikutuksen määrittelemiseen on käytössä kolme yleistä tapaa. Ensimmäinen tapa on kinemaattinen tapa, jossa tarkastellaan kappaleen etenemistä kohti pintaa. Kun huomataan, että kappale osuu pintaan, kappaleen nopeusvektorin osa, joka on samansuuntainen tason pinnan normaalin kanssa, vähennetään nopeudesta kahteen kertaan. Näin saadaan nopeuden suunta muutettua vastakkaiseksi. Yleensä käytetään myös vaimennuskerrointa, jotta liikkeestä saadaan realistisempi. Menetelmä on erityisen käyttökelpoinen partikkeleiden ja pallomaisten kappaleiden kanssa. Kuvissa 22 ja 23 on kuvattu nopeusvektorit pisteelle P ennen törmäystä ja törmäyshetkellä. [Parent, 2008]



Kuva 22.Pisteen P nopeuden V vektorit ennen törmäystä.

Pisteen P nopeus V törmäyksen jälkeen saadaan säilyttämällä tangentin suuntainen nopeuden tekijä, sekä vaihtamalla tason normaaliin nähden vastakkainen voima normaalin suuntaiseksi.



Kuva 23.Pisteen P nopeus V törmäyksen jälkeen

Toinen tapa määritellä törmäyksestä aiheutuva reaktio on kiinnittää nollanpituinen jousi törmäävästä kappaleesta pintaan ja antaa kappaleen läpäistä pinta. Metodissa toinen pinnoista on liikkumaton, jolloin törmäyksen vaikutuksista siihen ei tarvitse välittää. Jousi määrittelee pinnan normaalin suuntaisen voiman kappaleelle palata takaisin päin. Jousivoima vaikuttaa vain pinnan normaalin suuntaisesti, jolloin törmäävän kappaleen nopeusvektorin tangentin suuntainen komponentti ei muutu. Jousen käyttö määrittelemään törmäyksen aiheuttamaa reaktioita toimii myös monikulmioiden kanssa, mutta on monimutkaisempaa mahdollisten vääntävien voimien takia. [Parent, 2008]

Kolmas ja tarkin tapa määritellä törmäyksen vaikutus on laskea törmäyksen aiheuttama impulssi. Impulssi on voima, joka vaikuttaa hyvin lyhyen ajanhetken ajan. Impulssi vaikuttaa kappaleen momenttiin, niin lineaariseen kuin vääntömomenttiinkin. Näiden muutoksista saadaan laskettua lopulta muutos kappaleen nopeuteen ja kulmanopeuteen. Jos kappaleissa on useampi törmäyspiste, jokaisen törmäyksen vaikutus momentteihin on laskettava. Jokaisen momentin päivityksen jälkeen mahdolliset törmäyspisteet käydään uudestaan lävitse uusilla momenteilla niin kauan, kunnes uusia törmäyksiä ei enää havaita. [Parent, 2008]

6.2 Fysiikan rajoittaminen

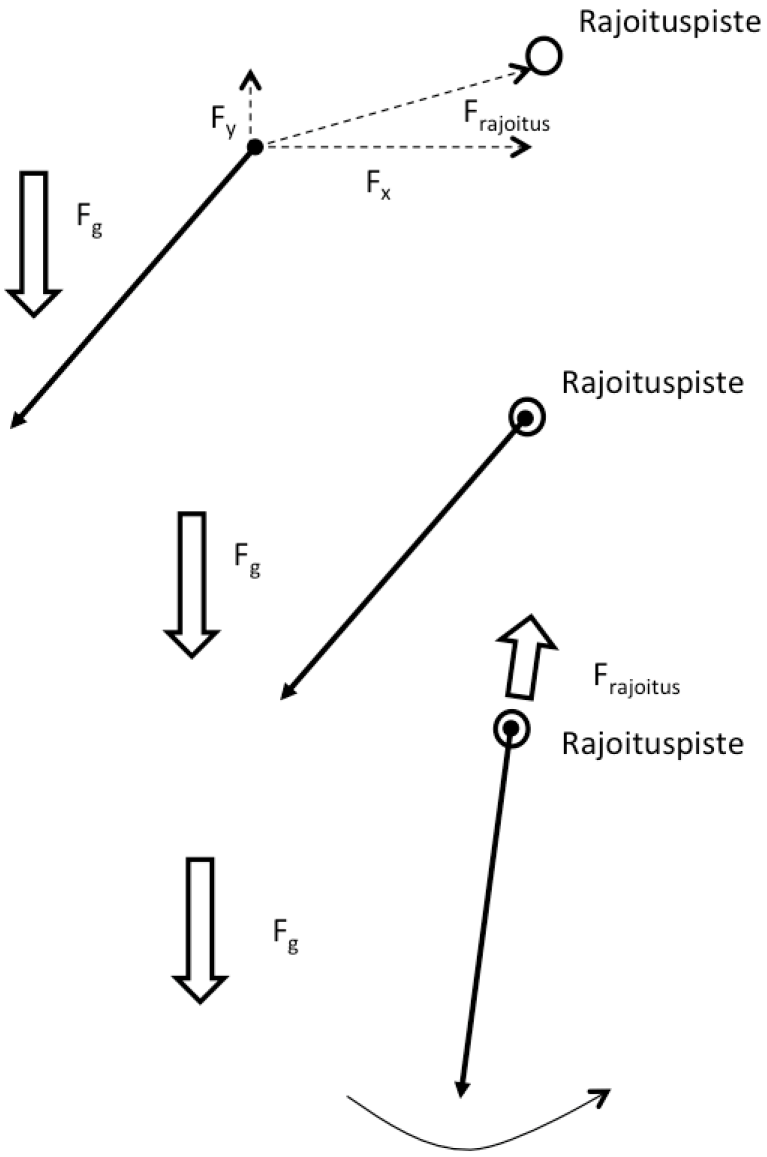
Sovellettaessa dynaamista liikkeenhallintaa nivelikkääseen olioon joudutaan usein tekemään rajoituksia ympäristön tai mallin fysiikkaan. Olion nivelissä jo itsessään saattaa olla rajoituksia liikeradoissa ja vapausasteissa, kuten esimerkiksi oikealla ihmiselläkin. Näiden lisäksi animaattori voi luoda keinotekoisia rajoituksia, kuten pienin mahdollinen etäisyys tai asento. Rajoituksia voi olla kahdenlaisia, kovia ja heikkoja. Kovat rajoitukset ovat sellaisia, että järjestelmä noudattaa niitä, kun taas heikon rajoitukset ovat sellaisia, että järjestelmä vain tekee parhaansa pyrkiäkseen noudattamaan niitä. Heikkoja rajoituksia käytetään järjestelmissä ulkopuolisten voimien lisänä vaikuttamaan liikkeeseen. Heikkoja rajoituksia ohjataan esimerkiksi energian minimointiongelman avulla. Jokainen rikottu heikko rajoitus lisää järjestelmän energiaa, kun järjestelmän on tarkoitus pyrkiä mahdollisimman pienienergiseen tilaan. Esimerkkinä käytetyistä heikoista rajoituksista on määritellä esimerkiksi jollekin objektin pisteelle nolla-energinen paikka koordinaatistossa. Tällainen rajoite houkuttelee systeemiä kuljettamaan objektin kyseiseen paikkaan, mutta mahdollistaa muutkin lopputilanteet. Rajoittamalla dynaamisesti tuotettua liikettä saadaan palautettua kontrollia liikkeestä takaisin animaattorille. [Parent, 2008] Dynaamisesti tuotetun liikkeen rajoittamista on käsitelty lisää kohdassa 7.4.

Kun ajatellaan, että hahmon koko liikerata on simuloinnin tulos, on selvää, että useamman sekunnin liikeradat muodostuvat laskennallisesti hyvin haastaviksi ja monimutkaisiksi. Siksi pidempää liikerataa luotaessa joudutaan yhdistelemään liike useammasta lyhyestä liikkeestä. Siksi esimerkiksi rajoittamalla pisteen paikkaa liikkeen loppuasennossa saadaan liikkeen jatkuvuus sulavamman näköiseksi. Liikesarjojen yhdistämistä on käsitelty tarkemmin luvussa 10.

6.3 Käänteisdynaaminen ongelma

Sen sijaan, että annetaan valmiiksi vaikuttava voima ja ohjataan liikettä sitä kautta, voidaan animoitavalle kappaleelle antaa rajoituksia pisteiden ja liikeratojen suhteen. Määrittelemällä kappaleelle haluttu piste avaruudessa algoritmin tehtäväksi jää laskea tarvittava voima, jotta kappale siirtyy pisteeseen ja täyttää annetun rajoituksen. Rajoituksen luonne on usein sellainen, että kun rajoitus on saavutettu, se myös pidetään muista voimista huolimatta. Kuvassa 24 on kuvattu heiluri avaruudessa, jolle määritellään

rajoitus pisteen muodossa. Rajoituksen saavuttamiseen tarvittavan voiman lisäksi heiluriin vaikuttaa painovoima.



Kuva 24.Käänteisdynaaminen ongelma.

Pisteen lisäksi annettu rajoitus voi olla myös sellainen polku pisteestä toiseen, joka kappaleen on kuljettava [Barzel and Barr, 1988]. Menetelmää, jossa lasketaan voima annettujen rajoitusten kautta, kutsutaan käänteisdynaamiseksi ongelmaksi.

6.4 Ajan ja avaruuden rajoitukset

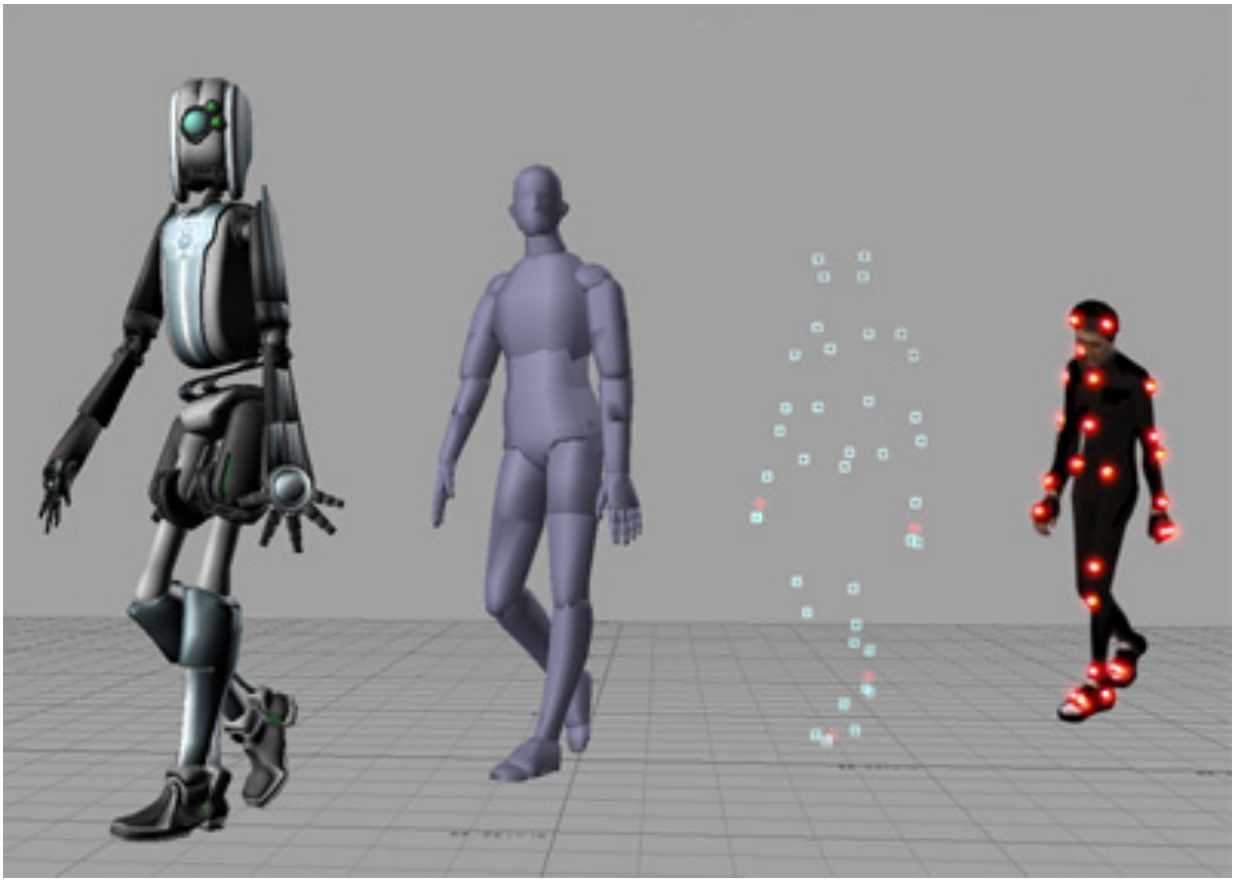
Yksi optimointimenetelmä, jolla on tarkoitus helpottaa halutun näköisen liikkeen luontia, on Witkin ja Kassin aika-avaruus-rajoitukset (Spacetime optimization) [Witkin and Kass, 1988]. Metodissa kappaleelle määritellään rajoituksia alkuasennon ja loppuasennon suhteen. Lisäksi määritellään muita rajoituksia, joilla voidaan vaikuttaa siihen, kuinka hyvin simuloitu liike noudattaa fysiikan lakeja. Algoritmi laskee liikeradan, joka kuluttaa vähiten energiaa, mutta samalla pysyy annettujen rajoitusten sisällä. Metodi sinänsä antaa animoijalle hyvän kontrollin animoitavaan kappaleeseen ja luo hyvin realistista liikettä, mutta menetelmä soveltuu huonosti monimutkaisten olioiden animoimiseen.

Abe ja muut [2004] esittelevät interpolointimenetelmän, jolla voidaan muokata olemassa olevaa liikesekvenssiä reaaliaikaisesti. Sen sijaan, että luodaan liikettä tyhjästä, heidän menetelmänsä on luoda uutta liikettä jo olemassa olevasta liikedatasta. Olemassa oleva liike voi olla dynaamisesti laskettu tai liikkeenkaappauksesta saatua dataa. Menetelmässä otetaan valmis sekvenssi, mutta liikettä muokataan aika-avaruus-rajoituksilla ja näin luodaan uusia liikeratoja. Uudet liikeradat lasketaan interpoloimalla alkuperäisistä liikeradoista.

7 Liikkeenkaappaus

Viimeinen käsitellyistä liikettä luovista liikkeenhallinta menetelmistä on liikkeenkaappaus (Motion Capture). Sitä käytetään animoinnin lisäksi myös reaaliaikaisen liikkeen generoinnissa. Jälkimmäinen näkyy esimerkiksi pelikonsolien myötä suosioon nousseessa liikkeen tunnistuksen käyttämisessä pelin ohjaukseen. Tällaisessa reaaliaikaisessa animoinnissa käytetään usein lisänä myös tallennettuja liikesarjoja. Käytettäessä liikkeenkaappausta animoinnissa liike yleensä tallennetaan erilaisilla sensoreilla, jotta sitä voidaan tarvittaessa uudelleen käyttää ja muokata. Kaapattuja liikkeitä voidaan myös yhdistää erilaisiksi sarjoiksi. Suurimmat haasteet liikkeenkaappaukselle asettavat fyysisten sensorien aiheuttamat rajoitukset. Sensoreiden välimatka lukijaan on aina rajattu ja liikkeen nopeudesta riippuen ongelmia saattaa aiheuttaa näytteenottotaajuus. Ongelmia pystytään osittain kiertämään langallisilla sensoreilla kuin myös nopeuttamalla näytteistettyä liikettä itse animointivaiheessa.

Kuvassa 25 on yksinkertaistettuna liikkeenkaappauksen eri vaiheet. Ensimmäisenä on näyttelijä, johon on kiinnitetty liikkeenkaappauksen vaatimat merkit, joita seuraamalla liike mallinnetaan. Tämän jälkeen liike voidaan toistaa esittämällä merkkien liikeradat. Kun merkkien liikerata on saatu sovitettua yksinkertaiseen hahmoon ja liike saatu näyttämään sujuvalta, voidaan mallin lopullinen tekstuuri lisätä.



Kuva 25. Liikkeenkaappaus merkeistä simuloituun malliin [wiki].

Liikkeenkaappausjärjestelmät voidaan karkeasti jakaa kolmeen luokkaan: mekaaniset, magneettiset ja optiset. [Thalmann and Thalmann, 1996]

Mekaaniset järjestelmät käyttävät erilaisia mekaanisia reaaliaikaisia syötelaitteita, esimerkiksi datahanskoja. Mekaanisissa järjestelmissä nivelten kulmat luetaan suoraan antureilta. Se, onko esimerkiksi hanskan sisällä käsi vai taivutellaanko nivelet asentoihinsa hanskan ulkopuolelta, ei vaikuta nivelten kulmiin. Mekaaniset järjestelmät soveltuvat monimutkaisen rakenteensa vuoksi huonosti esimerkiksi kokonaisen ihmisen liikeratojen kaappaukseen. [Thalmann and Thalmann, 1996]

Optiset järjestelmät pohjautuvat heijastaviin merkkeihin, joita on sijoitettu näyttelijän vartaloon tiettyihin pisteisiin. Seuraamalla näitä merkkejä voidaan tallentaa näyttelijän liike. Tallennusta voidaan suorittaa useammasta suunnasta samaan aikaan. Järjestelmän suuri etu on se, että näyttelijä voi liikkua vapaasti. Merkit ovat passiivisia komponentteja, jolloin dataa ei tarvitse liikuttaa näyttelijästä järjestelmään esimerkiksi kaapeleita pitkin.

Optisissa järjestelmissä ongelmia muodostavat esimerkiksi piiloon jäävät merkit. Tätä ongelmaa voidaan pienentää varmistamalla, että kohdetta kuvataan tarpeeksi monesta suunnasta. Tosin esimerkiksi kohteen alle jäävien merkkien seuraaminen on ongelmallista. Toinen ongelma tulee merkkien automaattisessa erottamisessa. Jos useampi merkki on esimerkiksi liikkeen seurauksena lähellä toisiaan, saattaa väärä tulkinta aiheuttaa ongelmia liikkeen mallinnuksessa. [Thalmann and Thalmann, 1996]

Magneettisessa järjestelmässä optiset merkit on korvattu sensoreilla, jotka mittaavat etäisyyttään ja asentoaan suhteessa keskitettyyn lähettimeen. Kun jokainen sensori on itsenäinen yksikkö, optisten järjestelmien kaltaista ongelmaa sensoreiden hetkittäisen läheisyyden kanssa ei ole. Suurin ongelma magneettisten sensorien käytössä on se, että kaikki sensorit tulee olla synkronoitu sekä keskenään että keskitetyn lähettimen kanssa. [Thalmann and Thalmann, 1996]

Optiset järjestelmät ovat kaikkein halvimpia tapoja tuottaa liikkeenkaappausdataa. Saadun datan jatkojalostaminen kaksiulotteisesta kuvasta kolmiulotteiseksi luurankomalliksi saattaa kuitenkin olla työlästä. Jokaisesta kuvatusta ruudusta pitää lukea kaikki merkit. Yksittäisen merkin lukeminen yksittäisestä ruudusta voi olla helppoa, mutta jos useampi merkki osuu lähelle toisiaan jossain ruudussa, on mahdollista, että merkit menevät jatkoruuduissa sekaisin ja saatu data kuvaa jotain muuta kuin tarkoitettua liikettä. Merkkien keskinäinen sekoaminen voidaan välttää käyttämällä merkkeinä esimerkiksi ledejä, jotka vilkuttavat omaa uniikkia koodiaan. Usein myös tarkasteltava merkki saattaa jäädä näyttelijän raajan taakse eikä näin ole näkyvillä useammassa peräkkäisessä ruudussa. Tällaisia ongelmia voidaan korjata interpoloimalla kyseisen merkin liikerataa. [Thalmann and Thalmann, 1996]

Kaapatun datan lähteestä riippumatta data pitää sovittaa esittävään malliin, jotta siitä saadaan luotua uutta liikettä. Sovittaminen manuaalisesti voi olla hyvin työlästä, siksi on kehitetty erilaisia menetelmiä sovittaa kaapattu data automaattisesti haluttuun malliin. Esimerkkinä voidaan mainita Kirk ja muut [Kirk et al., 2004], jotka käyttävät RANSAC-algoritmia sovittamaan kaapatun datan merkit simuloituun malliin. RANSAC-algoritmissa valitaan satunnainen joukko pisteitä, joihin sovitetaan malli animoitavasta hahmosta. Seuraavaksi lasketaan etäisyydet mallista muihin pisteisiin. Etäisyyden perusteella voidaan valita hyvät pisteet, jotka tallennetaan. Tämän jälken valitaan uusi

joukko pisteitä ja sama toistetaan. Se joukko, jossa on eniten hyviä pisteitä, valitaan ratkaisuksi [Fischler, 1981].

Liikkeenkaappauksen avulla voidaan tuottaa hyvin realistisen näköistä liikettä. Koska liike luodaan aina oikeasta datasta, liike saadaan näyttämään myös alkuperäiseltään. Esimerkiksi ihmisen kävelystä saadaan hyvin oikeannäköistä. Liikkeen kaappauksen rajoitteet tulevat kuitenkin esiin esimerkiksi siinä, että jokainen liike on näyteltävä erikseen, eikä sen avulla voida luoda keinotekoista liikettä. Liikkeen kaappauksesta saatua dataa täydennetäänkin usein muilla menetelmillä tuotetuilla liikkeillä, kun luodaan lopullisia liikesarjoja.

7.1 Merkittömän liikkeen kaappaus

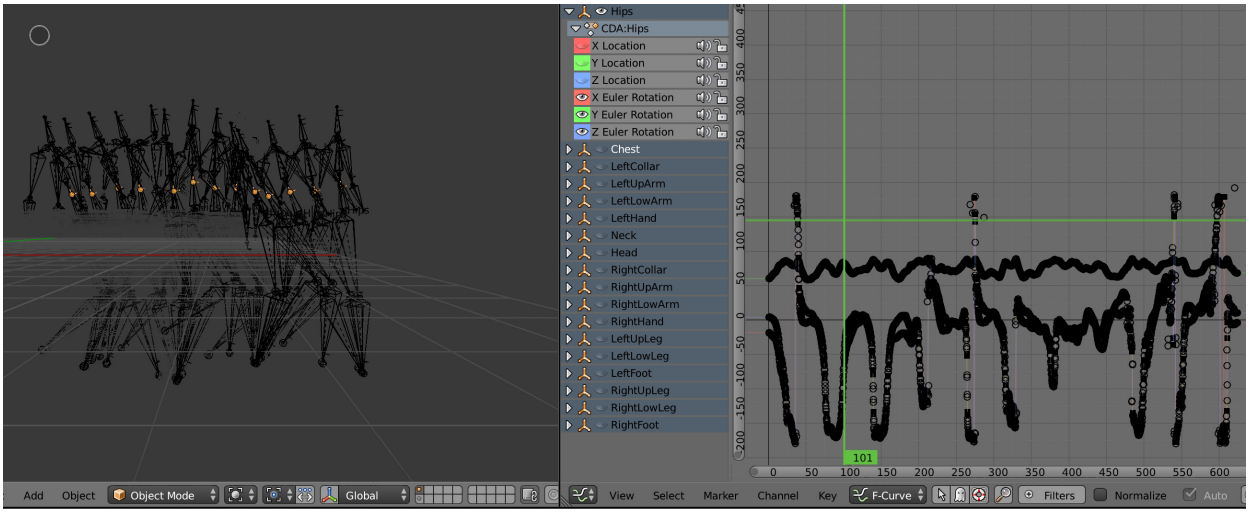
Viimeaikaisin liikkeenkaappaustutkimus on keskittynyt liikkeen tunnistamiseen ilman ulkopuolisia markkereita. Sovellus ei sinänsä tuo uutta liikkeenkaappausdatan käyttöön animaatiossa, mutta se avaa mahdollisuuksia esimerkiksi valvontakameroiden dataa tulkittaessa. [Gómez et al., 2012; Takahashi et al., 2010]

8 Liikkeen editointi

Olipa animaation liike tuotettu kinemaattisesti, dynaamisesti simuloimalla tai oikeasta liikkeestä mallintamalla, se vaatii tietokoneelta laskentatehoa. Monimutkainenkin liike kestää usein vain muutaman sekunnin, mutta kokonainen liikesarja paljon pidempään. Monimutkaisemman liikesarjan animointi edellä kuvatuilla tavoilla on usein liian monimutkaista ja laskennallisestikin mahdotonta. Siksi usein jo yksittäinen liike, esimerkiksi kamppailulajin liike, saattaa olla koostettu useammasta liikkeestä. Yksittäinen simuloitu liike liitetään toiseen simuloituun liikkeeseen. Jotta liike näyttäisi sujuvalta ja jatkuvalta, yksittäisiä liikkeitä joudutaan usein sovittamaan toisiinsa. Rajoittamalla esimerkiksi yksittäisen liikkeen loppuasennon koordinaatteja samoiksi seuraavan liikkeen alkukoordinaattien kanssa saadaan sovituksesta helpompaa.

8.1 Liikkeenkaappausdatan käsittely

Jotta liikkeenkaappauksella saadun datan avulla voidaan esittää haluttu liike uudestaan hierarkkisen mallin avulla, sitä tulee käsitellä. Saatu data on yksittäisten merkkien liikeratoja, joka näytteenottotaajuudesta ja näytteenottotavasta riippuen sisältää eri määrän virheitä ja kohinaa. Jotta data olisi uudelleenkäytettävissä, sitä tulee käsitellä ja mahdolliset virheet poistaa. Lopullisessa muodossaan data koostuu yleensä useasta kuvaajasta, jotka kuvaavat mallin eri parametrien arvoja ajan funktiona. Kuvassa 26 on nähtävissä esimerkki kävelevän ihmisen kuvankaappausdataa [blender]. Kuvan vasemmassa osassa on ihmisen luurankomalli ja oikealla on mallin lantion Euler-kulmien muutokset ajan funktiona.



Kuva 26. Kuvankaappaus Blender-ohjelmasta, johon on avattu kävelevästä ihmisestä kaapattu liike.

Liikkeenkaappausdataa voidaan käsitellä samoilla menetelmillä kuin muitakin analogisia signaaleja. Bruderlin ja Williams [1995] käsittelevät dataa taajuustasossa. Tavoitteena on sovittaa liikkeen raakadata haluttuun malliin. Brudelin ja Williams käyttävät peräkkäisiä ali- ja kaistanpäästösuodattimia, joilla dataa siivotaan ja yksinkertaistetaan. Menetelmässä muodostetaan saadusta signaalista ali- ja kaistanpäästö pyramidit, jossa jo kertaalleen suodatettu signaali suodatetaan uudestaan.

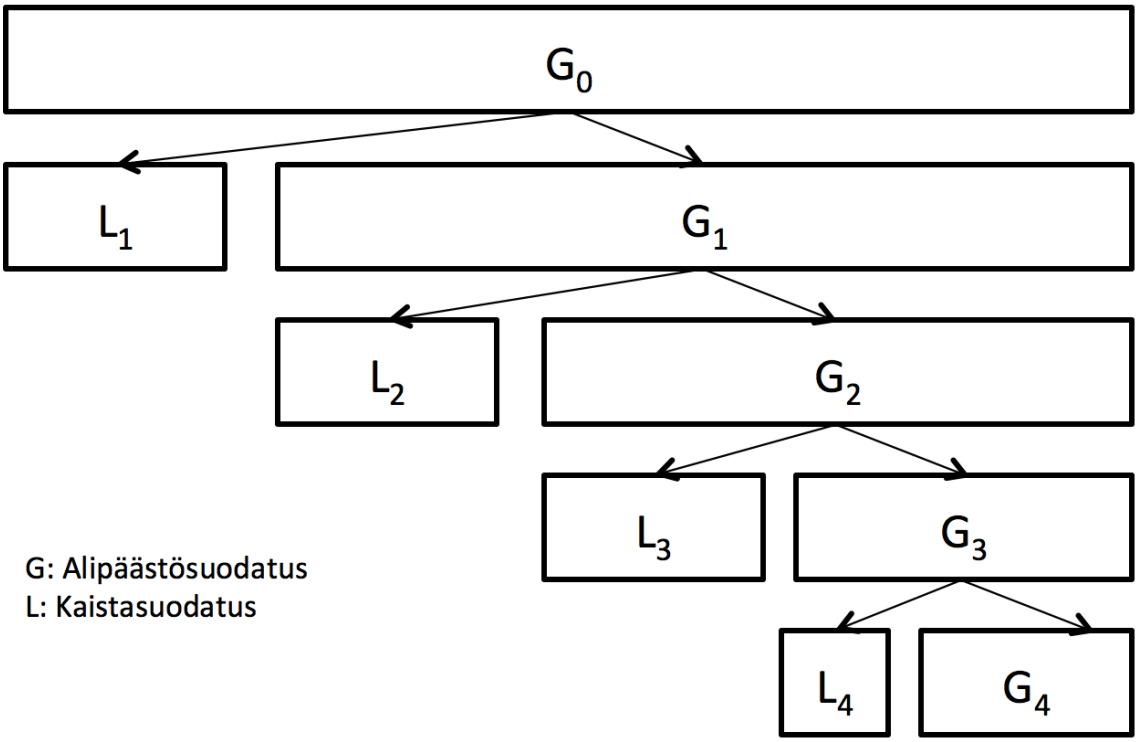
Kun G_0 on alkuperäinen signaali, L_n on kaistanpäästösuodatettu signaalin ja G_n on viimeinen alipäästösuodattu signaali saadaan

$$G_0 = L_1 + L_2 + L_3 + \cdots + L_{n-1} + G_n \, .$$

Uusi liike muodostetaan säätämällä eri taajuuksien voimakkuuksia kertoimella g_k ja kertomalla kaistanpäästösuodatettu signaali sillä

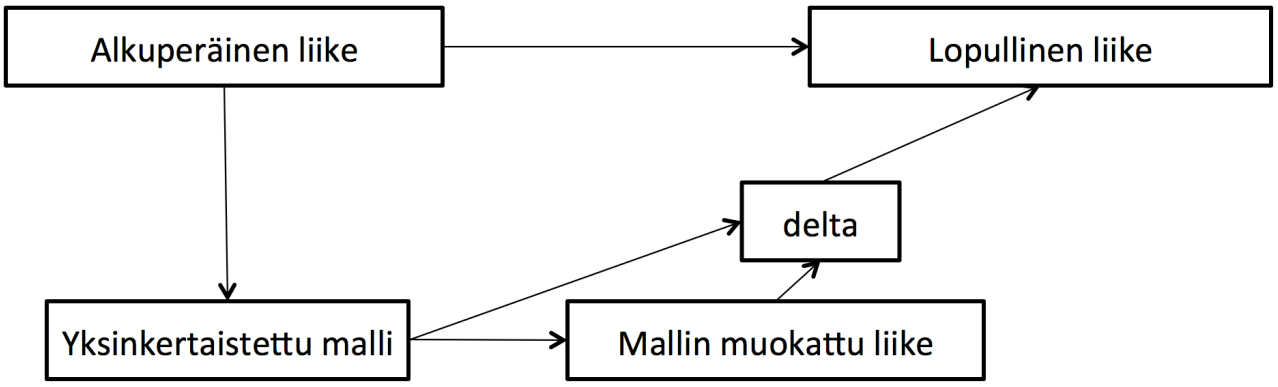
$$G_0 = G_{fb} + \sum_{k=0}^{fb-1} L_k \, ,$$

jossa fb = $\log_2$(m), kun m on signaalissa olevien kuvien määrä. Kuvassa 27 on esitetty peräkkäisen suodatuksen muodostama pyramidi.



Kuva 27.Peräkkäinen suodatus.

Taajustason käsittelyn sijaan signaalia voi käsitellä myös ajan funktiona. Witkin ja Popovic [1999] esittävät tavan muokata olemassa olevaa liikettä. Siinä liikettä kuvaavaa signaalia sovitetaan uudestaan aikatasossa. Algoritmisssa luodaan alkuun yksinkertaisin mahdollinen rajoitettu malli, jolla käsiteltävänä olevan liikkeen luonne saadaan esiin. Kun ensimmäinen sovitus on saatu aikaan, muokataan enää luotua mallia. Säätämällä vapausasteita, fysikaalisia ominaisuuksia ja paikkaa muodostetaan haluttu liike. Malliin tehdyt muutokset irrotetaan mallista ja nämä muutokset tehdään nyt alkuperäiseen dataan lopullisen liikkeen luomista varten. Witkinin ja Popovicin metodia on havainnollistettu kuvassa 28.



Kuva 28.Witkinin ja Popovicin tapa muokata liikettä mallin avulla.

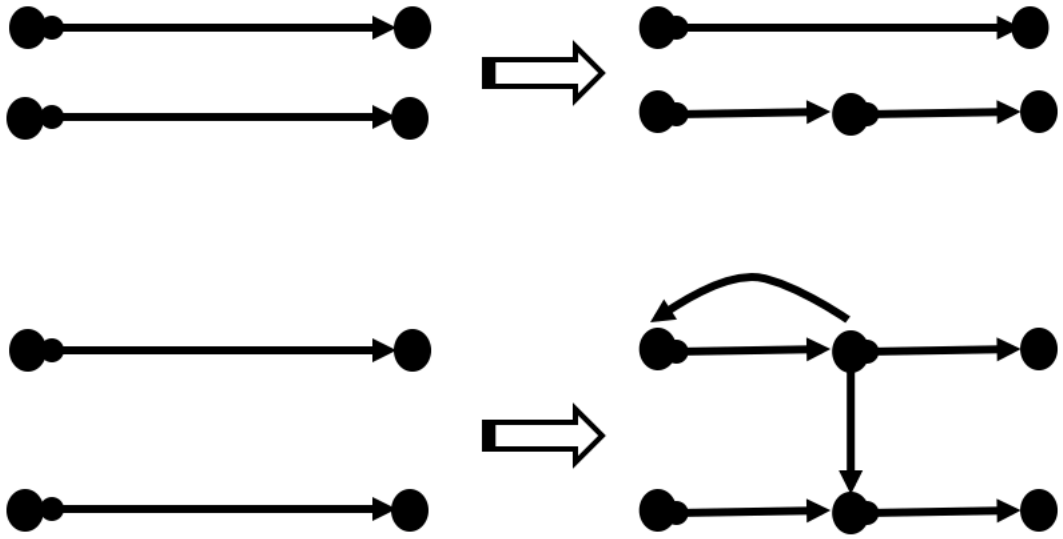
Menetelmän tekee työlääksi simuloidun mallin rakentaminen ja ensimmäinen datan sovitus. Mutta menetelmän hienous on siinä, että simuloitua mallia voi käyttää yhä uudestaan ja liikettä muokata tarpeen mukaan.

8.2 Liike kuvattuna graafissa

Yksi tapa yhdistää eri liikesekvenssejä on liikegraafit (Motion graph). Liikegraafit kehiteltiin auttamaan liikkeenkaappauksella saatujen liikkeiden yhdistämistä ja muokkaamista. Kovar ja muut [Kovar et al., 2002] kuvaavat tavan, jolla liikkeenkaappauksella luoduista liikesekvensseistä luodaan suunnattu graafi. Graafin avulla liikesarjoja voidaan järjestellä uudestaan ja luoda näin uusia liikesarjoja. Tavoitteena on löytää mahdollisimman luonnollinen siirtymä liikesarjasta toiseen.

Yksinkertainen liikegraafi luodaan lisäämällä kaikki liikesarjat verkon kaariksi luomalla näin 2n-solmuisen epäyhtenäisen verkon, jossa n on liikesarjan ensimmäinen tai viimeinen kuva. Liikesarjan keskelle voidaan myös lisätä solmu, jolla liikesarjasta muodostetaan kaksi erillistä kaarta. Solmukohdan lisäämisessä keskelle liikesarjaa on järkeä vain, jos on olemassa toinen liikesarja, jonka alku tai loppu on yhtenäinen solmuun tulevan tai siitä lähtevän liikkeen kanssa. Periaatteessa mikä tahansa kuva keskellä liikesarjaa voi olla solmu, johon saapuu alkuperäisen liikesarjan alun muodostama kaari ja josta lähtee sarjan loppuosan muodostama kaari. Koska uudet kaaret muodostuvat alun perin yhtenäisestä liikesarjasta, liike jatkuu sujuvasti kaaresta toiseen. Kuvassa 29 on esitetty kaksi yksinkertaista kahdesta liikesarjasta muodostettua graafia. Ylemmässä

esimerkissä vain toiseen sarjaan on lisätty solmukohta, josta lähtee vain yksi kaari, joka on kuvasarjan loppuosa. Alemmassa esimerkissä molempiin kuvasarjoihin on lisätty solmut, jotka mahdollistavat siirtymisen kuvasarjasta toiseen siirtymän avulla. Samassa esimerkissä on myös esitetty mahdollinen rekursiivinen paluu saman kuvasarjan alkuun.



Kuva 29.Kahdesta kuvasarjasta muodostettuja graafeja.

Usein kuvankaappauksella luodut liikesarjat eivät suoraan jatku siitä, mihin edellinen sarja päättyy, joten suora liittäminen saa aikaan epäluonnollisen liikkeen. Sovittamista varten tarvitaan erillinen siirtymäliikesarja. Olemassa olevasta liikkeenkaappauksella saadusta datasta etsitään mahdollisimman lähellä toisiaan olevia kuvia, joiden avulla siirtymä suoritetaan. Vaikka täysin toisiaan vastaavia kuvia ei löytyisikään, voidaan esimerkiksi kuvia yhdistämällä saada hyväksyttävä siirtymä aikaan. Se, kuinka hyvä siirtymän pitää olla, riippuu animoitavasta liikkeestä. Liialliset virheet katsojalle tutuissa liikkeessä, kuten kävelyssä, ovat luonnollisesti ei toivottuja.

Kovar ja muut [Kovar et al., 2002] esittelevät tavan valita sopivia kuvia siirtymien solmukohdiksi. Menetelmässä lasketaan datassa olevien nivelpisteiden etäisyydet jokaisen kuvan välillä ja valitaan lokaalit minimit mahdollisiksi siirtymäpisteiksi. Lokaali minimi ei tarkoita aina hyvää siirtymää ja siksi käyttäjältä odotetaan jonkinlaista raja-arvoa, jonka avulla siirtymät hyväksytään lopulliseen graafiin. Siirtymiä voidaan painottaa myös siten,

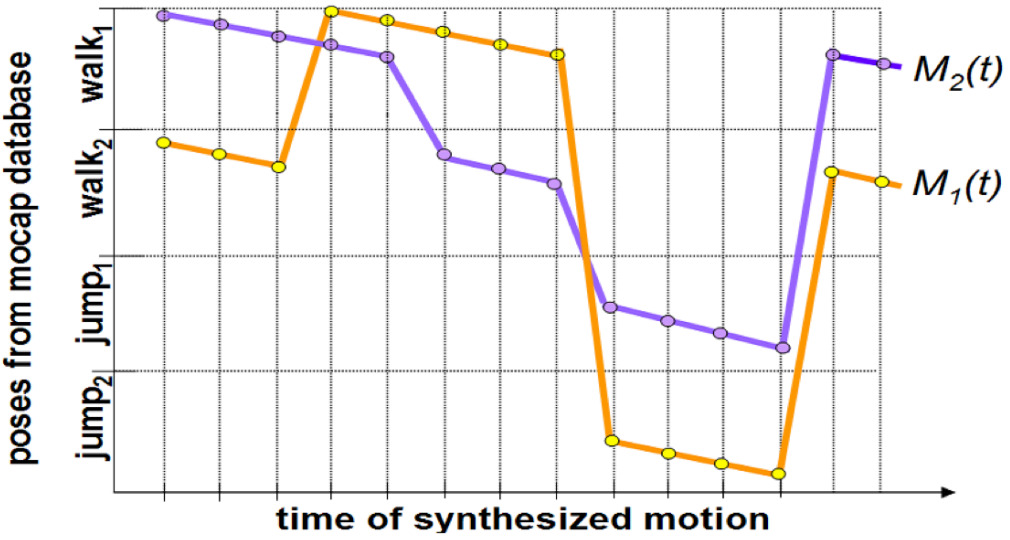
että kaaren seuraaminen ei kannata, jos kaaren seuraamisesta syntyy liikkeeseen epäjatkuvuuskohta. Painotuksilla algoritmi saadaan toimimaan paremmin myös edellä mainituissa tilanteissa, joissa tavoitteena on tuottaa tutumpaa liikettä kuten kävelyä.

Lopullinen liike luodaan kulkien graafissa ja lisäämällä kuljettujen kaarien sisältämät liikesarjat toisiinsa. Se, millainen liikesarja saadaan, riippuu luonnollisesti kuljetusta reitistä. Kovar ja muut [2002] käyttivät Branch-and-Bound-algoritmia etsimään polkuja, joilla saatiin yhdistettyä kaksi haluttua liikesarjaa toisiinsa. Raja-arvona he käyttivät aiemmin laskettuja nivelpisteiden etäisyyksiä.

Safonova ja Hodgins [2007] veivät liikegraafin käytön astetta pidemmälle. Heidän graafinsa muodostetaan muutoin samaan tapaan kuin Kovarin ja muiden graafi, mutta heidän graafin liikesarjat synkronoidaan ajallisesti, jotta kahden erillisen liikkeen välistä voidaan interpoloida uusi liike. Jos M1(t) ja M2(t) ovat polkuja, jotka muodostavat liikesarjan, saadaan interpoloitu liikesarja kaavasta

$$M'(t) = w(t)M1(t) + (1 - w(t))M2(t) ,$$

jossa w(t) on painokerroin, joka voi muuttua ajan myötä. Kuvassa 30 on neljän liikkeen sarja ja tästä muodostetut kaksi erillistä polkua. Lopullinen liike on jotain näiden kahden polun väliltä riippuen painokertoimesta w.



Kuva 30. Lopullinen liike saadaan interpoloimalla polkujen M_1 ja M_2 väliltä [Safonova and Hodgins, 2007].

Sofonova ja Hodgins tallentavat graafiin myös tiedon kappaleiden paikasta ja orientaatiosta, jota käytetään muun muassa laskettaessa eri siirtymille painoarvoja. Ennen hakua graafista karsitaan mahdollisia samanlaisia siirtymiä sisältäviä solmuja. Sofonova ja Hodgins käyttävät ARA*-algoritmia [Likhachev et al., 2003] etsittäessä optimiratkaisua interpolointi painotuksen ja polun suhteen [Safonova and Hodgins, 2007].

Myös Arikan ja Forsyth [2002] tutkivat graafien käyttöä. Algoritmi kulkee graafia ensin karkeammalla tasolla, jossa solmun kuvasarjan vierekkäisten kuvien oletetaan tiettyjen ehtojen täyttyessä olevan sama kaari. Tällöin graafia voidaan tiivistää huomattavasti ja laskentaa helpottaa. Tarpeeksi hyvää polkua etsittäessä polkua muutetaan korvaamalla kaaria vaihtoehtoisilla kaarilla tai laskemalla niiden painoarvoa. Uusien osapolkujen pitää täyttää annetut kovat rajoitukset, muuten ne hylätään. Jos muutoksen kautta ei saada aikaan uutta polkua, joka olisi parempi kuin olemassa oleva, katsotaan saavutetuksi lokaali minimi. Algoritmi pysäytetään, kun katsotaan, että tarpeeksi hyvä sarja on löydetty.

Arikan ja Forsyth muokkasivat lopullista liikettä haluttuun suuntaan käyttämällä rajoitteita. Rajoitus voi olla esimerkiksi solmu, jonka kautta polku puussa kulkee. Näin voidaan vaikuttaa kohteen asentoon tietyllä ajanhetkellä.

9 Yhteenveto

Tässä tutkielmassa on esitelty yleisimmät liikkeenhallintamenetelmät tietokoneanimaatiossa. Tutkielmassa on keskitytty käsittelemään liikkeenhallintaa ja sen ongelmia yleisemmällä tasolla. Samoja menetelmiä voidaan kuitenkin käyttää myös erikoistapauksissa.

Liikkeenhallinnassa käytetään vielä nykyäänkin varhaisista ajoista käytössä olleita menetelmiä. Välikuvien interpolointi avainkuvien väliin on käyttökelpoinen tapa tuottaa luonnolliselta näyttävää liikkuvaa kuvaa. Mutta avainkuva-animointi ei välttämättä ole paras valinta kaikkiin animaatiotarpeisiin. Kinemaattiset menetelmät sopivat hyvin, kun animoidaan esimerkiksi kirjoittavaa kättä. Sen sijaan dynaamisilla menetelmillä vastaavan animaation luominen on huomattavasti hankalampaa. Animoinnissa käytettäviä liikkeenhallintamenetelmiä ei myöskään voi verrata keskenään niiden hyvyyden perusteella. Menetelmä tulee valita tarpeen mukaan.

Eri liikkeenhallintamenetelmät ovat toinen toistaan täydentäviä. Kokoillan animaatioelokuvassa voi hyvinkin olla käytetty kaikkia tässä tutkielmassa käsiteltyjä liikkeenhallintamenetelmiä avainkuva-animaatiosta liikkeenkaappauksesta tuotettuun liikkeeseen.

Suuri osa elokuvien erikoistehosteista on tuotettu tietokoneella. Menetelmän suuria etuja onkin sen turvallisuus. Tietokoneella voidaan luoda liikkeenkaappauksen avulla liikutettavien olioiden lisäksi kokonaisia maailmoja, joiden rakentaminen puhtaasti lavasteiksi olisi mahdotonta. Tutkimus liikkeenhallinnan alueella onkin aktiivista johtuen todennäköisesti tietokonegrafiikan suuresta käytöstä esimerkiksi viihdeteollisuudessa. Suurin osa tutkimuksesta näyttää keskittyvät nimenomaan ihmisen animointiin liittyviin ongelmiin ja siihen, kuinka animaatiota voidaan tuottaa reaaliaikaisesti.

Viiteluettelo

[Abe et al., 2004] Yeuhi Abe , C. Karen Liu , Zoran Popović, Momentum-based parameterization of dynamic character motion, *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004 .

[Arikan and Forsyth, 2002] Okan Arikan , D. A. Forsyth, Interactive motion generation from examples, *ACM Transactions on Graphics (TOG)* 21, 3 (July 2002), 483-490.

[Barzel and Barr, 1988] Ronen Barzel , Alan H. Barr, A modeling system based on dynamic constraints, *ACM SIGGRAPH Computer Graphics* 22, 4 (1988), 179-188.

[BEO] Beowolf elokuva, http://www.imdb.com/title/tt0442933/ (1.9.2015).

[blender] http://www.centralsource.com/blender/bvh/iart/bvhfiles/dunk2.zip (25.8.2015).

[Bruderin and Williams, 1995] Armin Bruderlin , Lance Williams, Motion signal processing, *Proceedings of the 22nd Annual Conference on Computer graphics and Interactive Techniques*, 97-104, September 1995.

[Buss, 2009] Samuel R. Buss, Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and DampedLeast Squares methods, 2009, Available: http://www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf (21.9.2015).

[Fischler, 1981] M.A. Fischler, R.C. Bolles, RANSAC sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24, (1981), 381–395.

[Gómez et al., 2012] David Antonio Gómez, Jáuregui, Patrick Horain, Real-Time Particle Filtering with Heuristics for 3D Motion Capture by Monocular Vision, *ECCV'12 Proceedings of the 12th International Conference on Computer Vision - Volume Part III*, 663-666, 2012.

[Hecker et al., 2008] Chris Hecker , Bernd Raabe , Ryan W. Enslow , John DeWeese , Jordan Maynard , Kees van Prooijen, Real-time motion retargeting to highly varied user-created morphologies, *Proceedings of ACM SIGGRAPH 2008*, 2008.

[Huang, 1996] Zhiyong Huang, Motion Control for Human Animation, PhD Thesis, Swiss Federal Institute of Technology in Lausanne, Switzerland, 1996.

[Igarashi et al., 2005] T. Igarashi , T. Moscovich , J. F. Hughes, Spatial keyframing for performance-driven animation, *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, July 29-31, 2005.

[Kallmann, 2008] Marcelo Kallmann, Analytical inverse kinematics with body posture control, *Computer Animation and Virtual Worlds* 19, 2 (May 2008) , 79-91.

[Kirk et al., 2004] Adam Kirk , James F. O'Brien , David A. Forsyth, Skeletal parameter estimation from optical motion capture data, *ACM SIGGRAPH 2004 Sketches*, August 08-12, 2004.

[Kovar et al., 2002] Lucas Kovar , Michael Gleicher , Frédéric Pighin, Motion graphs, *ACM Transactions on Graphics (TOG)* 21, 3 (July 2002), 473-482.

[Lasseter, 1987] John Lasseter, Principles of Traditional Animation Applied to 3D Computer Animation, *Computer Graphics* 21, 4 (1987), 35-44.

[Likhachev et al., 2003] Likhachev, M., Gordon, G., and Thrun, S., ARA*: Anytime A* with provable bounds on sub-optimality, *Advances in Neural Information Processing Systems*, 767-774, MIT Press, 2003.

[Parent, 2008] Rick Parent, *Computer Animation: Algorithms and Techniques*, Morgan Kaufmann Publishers Inc, 2008.

[Popović and Witkin, 1999] Zoran Popović , Andrew Witkin, Physically based motion transformation, *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 11-20, July 1999.

[Safonova and Hodgins, 2007] Alla Safonova , Jessica K. Hodgins, Construction and optimal search of interpolated motion graphs, *Proceedings of the 2007 ACM SIGGRAPH*, 2007.

[Takahashi et al., 2010] Takahashi, K., Oida, T., Hori, J.-I., Hashimoto, M.,Remarks on markerless human motion capture using multiple images of 3D articulated human CG model, *Proceedings of the 18th European Signal Processing Conference*, 2010.

[Thalmann and Thalmann, 1996] Nadia Magnenat-Thalmann, Daniel Thalmann, Computer animation, *ACM Computing Surveys (CSUR)* 28, 1 (March 1996), 161-163

[Watt, 2000]Alan Watt, *3D Computer Graphics*, Addison-Wesley, 2000

[Witkin and Kass, 1988] Andrew Witkin , Michael Kass, Spacetime constraints, *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, 1988, 159-168.

[wiki] https://commons.wikimedia.org/wiki/File:Activemarker2.PNG. Checked 17.10.2015.

[Zhang and Liang, 2006] Hong Zhang, Y. Daniel Liang, *Computer Graphics Using JAVA 2D and 3D*, Pearson, 2006.